Facts and Reasons: Web Information Querying to Support Agents and Human Decision Making

Mehdi Samadi

CMU-CS-15-112

June 2015

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Manuel Blum, Co-Chair
Manuela Veloso, Co-Chair
Tom Mitchell
Craig Knoblock (USC/ISI)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research is sponsored through the Pradeep Sindhu Fellowship and the National Science Foundation (NSF) under Grants CCF-1101668 and CCR-0122581. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, the U.S. government or any other affiliated sponsor.

Keywords: Information Extraction, Information Validation, Knowledge Harvesting Systems, Web Mining, Knowledge Integration, Planning, Artificial Intelligence, Robotics

To my family, especially my sister, Samira.

Abstract

More than one million queries are made every minute on the Internet, and people are asking an ever increasing number of queries. Researchers have developed Information Extraction (IE) systems that are able to address some of these queries. IE systems automatically construct machine-readable knowledge bases by extracting structured knowledge from the Web. Most of these IE systems, however, are designed for *batch processing* and favor *high precision* (i.e., few false positives) over *high recall* (i.e., few false negatives). These IE systems have also been developed to readily evaluate only *factoid* queries (e.g., *what is capital of France?*). By contrast, many real-world applications, such as servicing knowledge requests from humans or automated agents, require broad coverage (high recall) and fast, yet customizable response times for *non-factoid* and *complex* queries (e.g., *Is shrimp meat healthy?*). Users may be willing to trade off time for accuracy. The existing IE techniques are inherently unsuitable to meet these requirements.

In this thesis, we investigate anytime applications, as information extraction tasks initiated as queries from either automated agents or humans. The thesis will introduce new models and approaches for learning to respond to the truth of facts using unstructured web information, while considering the credibility of sources of information.

We introduce OpenEval, a new *anytime information validation* technique that evaluates the truthfulness of knowledge statements. As input, agents or humans provide a set of queries that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect(Aspirin, GI Bleeding)*)), which the system should evaluate for truthfulness. OpenEval achieves high recall with acceptable precision by using *unstructured information* on the Web to validate information.

We extend the OpenEval approach to determine the response to a new query by *integrating opinions* from multiple knowledge harvesting systems. If a response is desired within a *specific time budget* (e.g., in less than 2 seconds), then only a subset of these resources can be queried. We propose a new method, AskWorld, which learns a policy that chooses which queries to send to which resources, by accommodating varying budget constraints that are available only at *query (test)* time. Through extensive experiments on real world datasets, we demonstrate AskWorld's capability in selecting the most informative resources to query within test-time constraints, resulting in improved performance compared to competitive baselines.

We further extend our information validation approaches to automatically measure and incorporate the *credibility* of different web information sources into their claim validation. To address this problem, we present ClaimEval, a novel and integrated approach which given a set of claims to validate, extracts a set of pro and con arguments from the Web

using the OpenEval approach, and jointly estimates the credibility of sources and the correctness of claims. ClaimEval uses Probabilistic Soft Logic (PSL), resulting in a flexible and principled framework which makes it easy to state and incorporate different forms of prior-knowledge. Through extensive experiments on real-world datasets, we demonstrate ClaimEval's capability in determining the validity of a set of claims, resulting in improved accuracy compared to state-of-the-art approaches.

Finally, we show how our information extraction techniques can be used to provide knowledge to anytime intelligent agents, in particular, for a find-deliver task in a real mobile robot (CoBot) and for a trip planner agent. We show that OpenEval enables robots to actively query the Web to learn new background knowledge about the physical environment. The robot generates the maximum-utility plan corresponding to a sequence of locations it should visit, asks humans for the object, and then carries it to the requested destination location. For the trip planner agent, we also contribute a novel method for a planner to actively query the open World Wide Web to acquire instant knowledge about the planning problem. We introduce a novel technique, called Open World Planner, that estimates the knowledge that is relevant to the initial state and the goal state of a planning problem, and then effectively generates corresponding queries to the Web using our OpenEval query system.

Acknowledgments

First and foremost, I would like to thank my advisors, Manuel Blum and Manuela Veloso, for everything they have done for me over the past 6 years. They helped me from the very beginning by believing in me, and patiently letting me explore a wide range of research topics. Their encouragement, help, and support, continued even after I left CMU to work on my startup. It has been an honor to work with both of you. The lessons that I have learned from you go far beyond the science, and will be with me for the rest of my life. THANK YOU.

I also want to thank my committee members, Tom Mitchell and Craig Knoblock, for serving on my thesis committee. I learned a lot from Tom in our individual meetings over the past five years as well as from his Machine Learning class, and Read the Web meetings. I appreciate the opportunity that I had to work closely with him. Craig's comments and advice have made this thesis stronger.

The faculties of the Computer Science and Machine Learning Departments have been a wonderful and supportive group. I would especially like to thank Luis von Ahn, Lenore Blum, Emma Brunskill, William Cohen, Roger Dannenberg, Mor Harchol-Balter, and Kayvon Fatahalian. I have also greatly benefited from the deep discussions and collaborations that I had with many people at CMU including: Partha Talukdar for our collaboration on AskWorld and ClaimEval projects; Thomas Kollar, Vittorio Perera, and Robin Soetens for our collaboration on the ObjectEval project; and Bryan Kisiel for helping me to integrate OpenEval into the NELL project.

Dave Mawhinney was extremely supportive over the past couple of years. He inspired me to pursue my dreams, helped me to push my research toward applying it to real-world problems, and introduced me to the entrepreneurial world.

I learned a lot about various robotics topics through Manuela's CORAL group: Joydeep Biswas, Susana Brandao, Brian Coltin, Max Korein, Tom Kollar, Som Liemhetcharat, Cetin Mericli, Tekin Mericli, JP Mendoza, Prashant Reddy, Stephanie Rosenthal, Yichao Sun, Junyun Tay, Felipe Trevizan, Rodrigo Ventura, Richard Wange, and Danny Zhu. I thank you for your support, and for providing constructive feedback about my research.

I am thankful to many of my friends for helping me through my life in Pittsburgh and for making my PhD life more enjoyable. I am thankful to my family for their love, support, and for confidence in me. I left them seven years ago in order to continue my studies. They have been actively supporting me since then. This dissertation would not have been possible without them.

Contents

1 Introduction **1**

1.1 Thesis Approach 4

1.1.1 Anytime Approach for Web Information Querying 5

1.1.2 Knowledge Integration and Time-Budgeted Query Answering . . 5

1.1.3 Measuring Credibility of Sources and Incorporating the Feedback 6

1.1.4 Combining Retrieval and Use of Knowledge 7

1.2 Contributions 8

1.3 Evaluation 10

1.4 Document Outline 10

2 Anytime Approach for Web Information Query Evaluation **13**

2.1 Introduction 14

2.2 OpenEval 17

2.2.1 Context-Based Instance (CBI) Extractor 19

2.2.2 Learning 27

2.2.3 Predicate Instance Evaluator 33

2.3 Experimental Evaluation 37

2.3.1 Setup 38

2.3.2 Comparison Metrics 39

2.3.3 Baseline Approaches 39

2.3.4 Accuracy of OpenEval in terms of Precision and Recall 41

2.3.5 Comparison of Different Values of Parameters 42
2.3.6 Results of Using OpenEval in Different Applications 43
2.3.7 Using OpenEval as part of the NELL System 49
2.4 Summary 53

3 Knowledge Integration and On-Demand Time-Budgeted Query Answering 55
3.1 Introduction 56
3.1.1 Requirements that a Knowledge-On-Demand Systems Should Accommodate 57
3.1.2 AskWorld 58
3.2 Problem Statement 60
3.3 Our Approach 61
3.3.1 Markov Decision Process (MDP) Formulation of BUDGET POLICY EXECUTOR (BPE) 61
3.3.2 Solving MDP 65
3.3.3 KNOWLEDGE INTEGRATOR 69
3.4 Experimental Results 69
3.4.1 Datasets & Setup 69
3.4.2 Does polling KRs for non-query predicates help? 71
3.4.3 Are budget-sensitive policies able to select effective polling queries? 72
3.5 Summary 78

4 Measuring Credibility of Sources and Extracting Reasons 79
4.1 Introduction 80
4.2 Our Approach: ClaimEval 83
4.2.1 Credibility Assessment (CA) Graph Construction 84
4.2.2 Joint Source Credibility Estimation & Claim Evaluation 88
4.2.3 Learning Rules Weights 94
4.2.4 Examples 94
4.3 Experimental Evaluation 98

4.3.1 Setup . . . 98
4.3.2 Baseline Approaches . . . 101
4.3.3 Comparison to Baselines: Example . . . 104
4.3.4 Comparison to Baselines: Experimental Results . . . 107
4.4 Summary . . . 108

5 Using the Web to Interactively Learn to Find Objects 111
5.1 Introduction . . . 112
5.2 ObjectEval . . . 114
5.2.1 Model . . . 115
5.2.2 Querying the Web using OpenEval . . . 117
5.2.3 Inferring a Plan . . . 119
5.3 Evaluation . . . 119
5.3.1 Predicting the Location of Objects . . . 119
5.3.2 Simulated Experiments . . . 121
5.3.3 Robot Experiments . . . 124
5.4 Summary . . . 127

6 Iterative Query-Based Open World Planning 129
6.1 Introduction . . . 130
6.2 OpenWorld Planner . . . 132
6.2.1 Definitions and Notations . . . 133
6.2.2 Constructing Knowledge Graph . . . 134
6.2.3 Layer-by-layer construction of knowledge graph: . . . 135
6.2.4 Example of knowledge graph . . . 136
6.2.5 Prioritizing constraints . . . 137
6.2.6 Instantiating Constraints . . . 140
6.2.7 Querying OpenEval and Calling Downward Planner . . . 141
6.3 Experimental Results . . . 141

6.4 Summary 143

7 Related Work **145**

7.1 Use of Co-Occurrence Statistics in Information Extraction 145

7.2 Extracting Factual Information from Unstructured Web Text 147

7.3 Generic Relation Extraction 149

7.3.1 Relation Extraction Using Wikipedia 151

7.4 Credibility of Information Sources 152

7.5 Searching for Objects 157

7.6 Integration of Robots with the Web 158

7.7 Retrieving the Information from the Web for Planning 158

7.8 Budget-Sensitive Query Evaluation 160

7.9 How Our Work Fits 163

8 Conclusion **165**

8.1 Thesis Contributions 165

8.2 Future Research Directions 168

8.3 Concluding Remarks 171

A Domains used in the Open World Planner **173**

A.1 Domain 1 173

A.2 Domain 2 174

A.3 Domain 3 176

Bibliography **181**

List of Figures

2.1 The architecture of OpenEval. OpenEval consists of three main components: Learning, Predicate Instance Evaluator, and Context-Based Instance (CBI) Extractor. 18

2.2 An example of the text that is found for predicate *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*. 20

2.3 An example of a webpage that is retrieved for the query *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*. The solid line is drawn manually and shows the main information block of the webpage. Other information blocks in the webpage are non-informative and should be mostly removed during the extraction. 22

2.4 A DOM tree representation of part of a webpage that is retrieved for the query *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*. For the snippet *"Common Aspirin Side Effects: The common aspirin side effects are..."*, the text of the snippet is split between different DOM nodes in the tree, e.g. the word "aspirin" is separated from other parts of the snippet since it is represented by the anchor tag node **a**. 23

2.5 Entropy values (value of E in Equation 2.1) for three different cases. The first figure from the left shows that the entropy value is high since the classifier has classified most of the (positive) instances in the set CBI as negative. The figure in the middle shows that the entropy value is equal to zero (the classifier is performing almost randomly). The figure on the right shows that the entropy value is low since the classifier is able to classify most of the instances correctly (classifying instances in CBI as positive). 29

2.6 The top 20 highest-ranked features that are extracted for predicate *Religion(x)*. The first row shows the features that are learned by the classifier using only two predicates *Religion(x)* and *AcademicField(x)* for training. The second row shows the learned features when all the predicates in Tables 2.1 and 2.2 are used in training. In general, having more predicates in the ontology would improve the accuracy of OpenEval, since a more diverse set of negative examples would be chosen from the ontology. . . 31

2.7 The top 20 highest-ranked features that are learned for predicate *AcademicField(x)*. The number of seed examples that are used in the training is shown on the left. 33

2.8 The precision/recall curve for test predicate instances. OpenEval uses 100 iterations for training and 5 iterations for evaluation. We also plot the curve for two baseline approaches. 41

2.9 F1-score of OpenEval when it uses different numbers of iterations for training. 43

2.10 Precision, recall, and F1-score of OpenEval when it uses different numbers of iterations for evaluation. 43

2.11 List of location types used to train and test OpenEval. 44

2.12 The precision/recall curve of the OpenEval approach for over 40 test locations. 45

2.13 The precision/recall curve of the OpenEval approach for predicates *PicturableWord(x)* and *NonPicturableWord(x)*. 47

2.14 The precision/recall curve of the OpenEval approach for approved and withdrawn drugs. The test data consists of 12 approved and 12 withdrawn drugs. The withdrawn drugs are listed as "approved" in the DrugBank database 2013 [Knox et al., 2011], but are withdrawn by official sources (these are errors in the database). 48

2.15 NELL's software architecture. Reprinted from [Mitchell et al., 2015a]. . . 50

2.16 Predicates (slot names) used for entity person in KBP 2013 evaluation. . 51

2.17 Predicates (slot names) used for entity organization in KBP 2013 evaluation. 51

2.18 Overview of the CMUML Slot Filling (SF) system. Reprinted from [Bryan Kisiel, 2013]. 52

3.1 Architecture of the AskWorld system. Given a user query (e.g., *UnHealthyFood(sugar)?*) and a time-budget within which this query must be answered (e.g., 2sec above), the Budget Policy Executor selects a subset of the available knowledge resources (KRs) to poll (shown by solid lines in the figure; dotted lines connect knowledge resources that are not used to answer this particular query at the specified time budget). Responses from the KRs are integrated by the Knowledge Integrator and the final response is returned to the user within the specified time budget. 59

3.2 List of the 25 categories used in our experiments. For each category, we use 200 training and 50 test instances. 70

3.3 Precision-Recall curve comparing performance of AskWorld when it is restricted to issue queries only related to the target predicate (*Target Predicate Queries*) vs. when it is allowed to issue queries for all predicates in the ontology (*All Predicates Queries*). All results are averaged over all 25 target predicates. We observe that allowing more diverse queries, as in the *All Predicates Queries* setting, significantly improves performance. . . . 72

3.4 F1 scores comparing different systems against varying test-time budgets (in milliseconds). We observe that variants of the proposed approach, AskWorld, outperform all other baselines. 75

3.5 Three examples of policies learned by the AskWorld (V*) system. For each example, the input query to the system is first shown (with grey background), followed by the ordered list of queries asked by AskWorld. For each query asked by AskWorld, we show the name of the resource that is queried (in bold), the name of the predicate, the cost of the query (in milliseconds), the prediction score returned by the resource, and the leftover budget. In each case, AskWorld predicts *true*, the correct response. . . . 77

4.1 The Credibility Assessment (CA) Graph is constructed to evaluate C2, while the user has already specified the other claim C1 to be true (green concentric rectangle). In addition to the claims, the CA graph also consists of nodes corresponding to *Domains*, *Webpages* and *Evidences*. While some evidences are supportive of the claims (solid green lines), others are not (red dotted line) (see Section 4.2.1 for details). ClaimEval, the proposed system, estimates the credibility of the domains based on the assessment available on claim C1, and combines that with the level of support from evidences originating from those domains to evaluate claim C2 (see Section 4.2.2 for more details). 82

4.2 Five examples of the Credibility Assessment Graph, with the credibility score assigned to each node in the graph. Similar to Figure 4.1, the graphs' layers from the left are: *Domains*, *Webpages*, *Evidences*, and *Claims*. A redline connecting an evidence to a claim is an indicator of a *con* argument, while a *green* line is an indicator of a pro argument. White nodes are the nodes where the credibility value is not known, and a node with the green color means that it is credible. A double-lined circle indicates the training data (i.e., nodes which we know the truth value). 95

4.3 Scores assigned by ClaimEval where we use different values for the weight of training data (denoted by W). 97

4.4 Step by step demonstration of how a Credibility Assessment graph is converted to a bipartite graph, which is then used as an input of baseline fact-finding algorithms. 102

4.5 (a) An example of a Credibility Assessment graph, (b) and its equivalent bipartite graph. 105

4.6 An example CA graph, with the scores assigned by different approaches marked on top of each node. While baselines such as Majority Vote, Generalized Sums and AverageLog overfit by over-trusting domains with many evidences (e.g., node 13), ClaimEval is able to find the annotator judgments. 106

4.7 Performance of different systems when increasing amounts of evidence are available. ClaimEval, the proposed system (top-most plot), is best able to exploit additional evidence achieving the best overall performance. . . 108

5.1 Examples of snippets returned by search queries for the predicate locationHasObject. On the left are some instances of the predicate objectInLocation that we have queried the system with. In the middle, is the associated probability of this predicate instance according to OpenEval. On the right, are two documents returned by a web search that are used by OpenEval to evaluate the probability of the predicate. 113

5.2 A robotic platform that we have used to find and fetch objects. 114

5.3 An example of our robot searching for an object. In (a) the system gets a query to find a "coffee" and take it to room 7001. In (b) it goes to the nearest kitchen. In (c), it asks a person to place a coffee on it. In (d), it gets the coffee and the person says that the robot has the object. In (e), the robot delivers the object to its destination. 115

5.4 An example of how ObjectEval queries the Web to find the maximum probability location of "papers." In this example, two location types exist: printer room and kitchen. A Web search for "papers," "printer room" and "papers," "kitchen" returns a set of webpages from which text snippets are extracted. The features of these text snippets are then categorized as one of the location types. The red border indicates that the text snippet was categorized as a "kitchen" and the green border indicates it was categorized as a "printer room." The frequency of the resulting categorization is then used to compute the maximum likelihood probability of each location given a query object. 118

5.5 The precision/recall curve of OpenEval for the 45 test predicate instances for location types: *bathroom*, *printer room*, *kitchen*, and *office*. 121

5.6 The F1-score for the 45 test predicate instances when training on a subset of the training dataset. The figure shows that OpenEval achieves a high F1 score even when it uses a few training examples, and its F1 score significantly increases when more training examples are provided for training. . 122

5.7 The number of locations visited by the robot before finding the query object for the interactive mode of ObjectEval (red line) and the baseline (green line). The data is sorted by the number of visited locations per simulation run. 125

5.8 An example of interaction between a human and CoBot. CoBot queries OpenEval in order to retrieve the information that it needs to execute the required task (e.g., determining where it is likely to find object "coffee"). 127

6.1 Simplified version of trip planning problem (not the actual PDDL format). 132

6.2 Part of the knowledge graph built for the simplified version of the trip-planning domain. 137

6.3 Examples of constraints that are extracted for action *Book-Hotel*. 138

6.4 An example of a plan found by Open World Planner for traveling from *Boston* to *Chicago* while visiting an arbitrary city (*Austin*) in between. . . 143

List of Tables

2.1 List of categories (predicates with one argument) used to train and test OpenEval. 38

2.2 List of relations (predicates with two arguments) used to train and test OpenEval. 39

2.3 Official evaluation scores of various CMUML submissions. *CMUML* is the system based on NELL, and *CMUML+OpenEval* uses OpenEval as the post-processing validation step. 52

4.1 First-order logic rules that define how the information flows between an evidence and a claim. 90

4.2 First-order logic rules that define how information flows between a source and an evidence. 91

4.3 First-order logic rules that define how information flows between a domain and a source. 91

4.4 The accuracy of the Majority Vote (MV), Generalized Sums (GS), TruthFinder (TF), Average-Log (AL), Generalized Investment (GI), Pooled-Investment (PI), and ClaimEval (CE) techniques in predicting the truth values for different categories of claims. The maximum value of each row is shown in bold. ClaimEval, the proposed system, achieves the best overall performance. 107

5.1 The probability that ObjectEval assigns to different test objects for each location type. The location type with the maximum probability is shown as bold. 120

5.2 Average and standard error for the number of visited locations, distance and number of interactions for ObjectEval and baseline approaches. The baseline uses only the terms for interaction I and distance D from Equation 5.2. ObjectEval (offline) uses batch training and ObjectEval (interactive) is given no training data, but instead uses the presence of objects in locations to update the probability of a location given the object as it performs a search (as from Equation 5.2). 123

6.1 Total number of queries sent to the Web (OpenEval) using Open World Planner and the baseline approach that instantiates all of the predicate instances. The table also shows the accuracy of plans found by OWP. . . . 142

7.1 Comparison of characteristics of recently proposed feature-cost sensitive learning methods. The method presented in this thesis is the only one which is both test-time budget sensitive, and is flexible enough to handle multiple test-time budgets without retraining. 160

List of Algorithms

1 OpenEval - Context-Based Instance (CBI) Extractor 19
2 OpenEval - Traversing and processing information of the DOM tree in the depth-first order 25
3 OpenEval - Parsing a DOM node 26
4 OpenEval - Learning classifiers 27
5 OpenEval - Predicate instance evaluator 36
6 AskWorld - Query evaluation for knowledge-on-demand 62
7 ClaimEval - Evaluating the correctness of a alaim 84
8 ClaimEval - Building the Credibility Assessment graph 86
9 Open World Planner 133

Chapter 1

Introduction

The World Wide Web (WWW) is the most diverse and largest source of information that exists today, and it is expected to grow even more as the world's primary source of knowledge over the next decade. This rise of the Web has made it possible for both humans and machines to get access to over 4.9 billion webpages. Since an estimated 80% of all of the information currently on the Web is *unstructured* (e.g., webpages and blogs), sophisticated information extraction techniques are required to transform such noisy content into usable data. Even with *structured* data, contextualization and extraction of the right data remains a challenge.

Search engines, such as Google, are currently used to retrieve information from either structured or unstructured pages on the Web. Most of these search engines return results in a multitude of links to webpages deemed relevant to the search query; however, for the most part, they do not provide any support in processing the information. For example, in order to decide whether a given claim is true or false, one is forced to sift through a long list of URLs, identify a set of relevant documents among different sources of information on the Web, analyze the content of these noisy documents, and aggregate the information. The process can be error-prone, piecemeal, inconsistent, and tedious. The Web search engines address only part of the overall problem, viz., producing only a list of relevant sources.

Processing the retrieved information is important when a human or an external applica-

tion (e.g., an agent) wants to retrieve specific information embedded in the search results. For an individual, the process of reading the content of multiple pages, understanding their content, and aggregating this surplus of information is an inefficient task. For an agent, retrieving Web information and embedding the retrieved information with the task that it is performing is a challenging problem. Moreover, it is practically impossible to manually process thousands or millions of websites that are returned by the search engines.

A wide variety of complementary trends have started changing this transformation of the Web, from providing a set of links, to also processing the information and providing structured data that can be easier to use for both human and external applications. Large-scale question-answering systems such as Watson [Ferrucci et al., 2013] and information extraction systems such as Google Knowledge Graph [Google, 2012] are two examples of these attempts at structuring data for more efficient access and utility. Other significant contributions have been made by automatically constructing machine-interpretable knowledge bases by extracting relational facts and rules from large text corpora such as the Web (e.g., NELL [Carlson et al., 2010b]). These techniques aim to automate the process of sifting through URLs returned by search engines, and retrieve an answer in response to the user's question, hence reducing what once took hours into a task that only needs a few minutes.

Despite the wide success of the current Information Extraction (IE) techniques, most of them are designed for *offline batch processing* and for ensuring high precision (i.e., few false positives) knowledge extraction. This high precision may result in low recall (i.e., few false negatives), making them less applicable to applications that require *high recall*. Moreover, in many applications, such as mobile-robot control, the *anytime* capability is required, where a human or an agent expects that the quality of results improves gradually as computation time increases.

The *anytime aspect* and *high recall* are especially important when a human or an external application, such as an automated agent, wants *to query* the knowledge base and expects to get responses as soon as possible. For example, a mobile robot may need to find an object (e.g., coffee) in an environment for which it requires to know the location type (e.g., kitchen) of the input object. The knowledge acquisition technique that the robot is using should provide the location of the input object and should also cover different

locations and objects for which the robot may query the knowledge base.

In addition to *anytime* capability, some applications may require the response to be provided within a specific time budget (e.g., in less than 2 seconds). Depending on the task that the robot/agent is performing, it may need different types of knowledge (instances of different predicates), and responses within different time budgets. If a response is desired within a time budget, then perhaps only a subset of information resources on the Web can be retrieved and processed. There is a need for an automated *knowledge-on-demand* and *budget-sensitive* technique to choose which queries to send to which resources and how to retrieve and process the information. Such a technique should be able to learn a policy that accommodates varying budget constraints that are available only at query (test) time. Ideally such a technique should be able to incorporate an on-demand *knowledge integrator* that aggregates opinions not only from text data on the Web, but also from other types of knowledge resources such as different existing KBs (e.g., YAGO), information extraction systems, and subcomponents from existing knowledge harvesting tools such as NELL.

Another major challenge of retrieving knowledge from the Web is the fact that the Web, by its nature, is a large and open system in which anyone may contribute. The open and collaborative nature of the Web raises the question of how much a given source of knowledge can be *trusted*. The current knowledge base (KB) systems are centered upon extracting factoid knowledge, e.g. *"Paris is the capital of France"*). However, evaluating the correctness of non-factoid *claims* (e.g., *"Shrimp meat is healthy"*), is particularly a challenging problem as two different webpages may contain conflicting evidences even related to a single claim. For example, while an animal rights website might not support meat-eating and thus term turkey meat as unhealthy, the website of a grocery store might claim otherwise. Additionally, a scientific paper focusing on this question might provide the most authoritative answer. One would want to trust evidences contained in the credible source (the scientific paper) and ignore the other two. Hence, given a set of claims, one needs to identify relevant sources on the Web, extract supporting and contradictory evidences from those sources, estimate source credibility, and finally aggregate all of this information to evaluate the given set of claims. This is a time consuming process, and thus, there is a growing need for an *integrated* approach for automatic extraction of relevant evidences and sources, estimation of information source credibility and utilizing

those credibility estimates in claim evaluation.

In this thesis, we address the above challenges in the context of solving a specific sub-problem in knowledge acquisition – *how to tell whether a given claim is (likely to be) true or false*. We contribute techniques that take advantage of the powerful corpus of the Web data in order to evaluate the truth of a claim. Evaluating truth of a claim would have many applications. For example, many popular questions that have been searched in Google (e.g., "is yogurt healthy?", "is zero a number?") can be directly converted to propositions that require assessment of their correct value. Moreover, in applications such as automated question answering, most of the questions and their candidate answers can be converted to a proposition that requires assessment. For example the question "where is Beijing located?" and the corresponding answer "China" can be converted to a proposition "Beijing is located in China". It has been also shown that evaluating correctness of a proposition can be used in question answering techniques [Harabagiu and Hickl, 2006; Magnini et al., 2002], information extraction systems [Etzioni et al., 2004, 2008; Carlson et al., 2010d], and robotics [Samadi et al., 2012].

The principle question addressed in this thesis is:

> How can we build budget-sensitive, knowledge-on-demand models for jointly estimating the credibility of sources and the validity of queried claims using unstructured web information?

In this thesis, we will address the above challenges by investigating anytime applications, as information extraction tasks initiated as *queries* from either *automated agents* or *humans*.

1.1 Thesis Approach

This thesis has four major contributions. First, we contribute a novel anytime IE technique to respond to predicate-based queries while also achieving high recall (at a small cost of sacrificing precision). Second, our IE technique is extended to integrate the answers provided by any other types of IE systems, and also to provide an on-demand time-budgeted

query answering service. Third, we investigate the joint estimation of credibility of sources and correctness of claims. Our technique also allows incorporating and propagating information related to feedback that may be collected from either a human or from the use of the retrieved information by an automated agent. We show that such feedback can be used to improve the accuracy of our knowledge-providing technique. Finally, we apply our IE technique to two automated agents, where each agent automatically generates queries based on its missing and needed knowledge. We also show how the retrieved knowledge is combined with the task that the agent is performing. In the next sub-sections, we briefly explain each of these components.

1.1.1 Anytime Approach for Web Information Querying

In this thesis, we first introduce OpenEval, a new online and anytime information validation technique, which uses information on the Web to automatically evaluate the truth of queries that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect (Aspirin, GI Bleeding)*)). OpenEval gets a small number of instances of the predicate as seed positive examples. OpenEval automatically converts them into a set of training data by querying the Web and processing unstructured webpages retrieved by the query. Each resulting training is a bag-of-words extracted from the context surrounding the given predicate instances on the webpages. These context-based training instances are then used as positive examples to train one classifier for the given predicate, while a sample of the training instances of other predicates are used as negative examples. To evaluate a new instance of a predicate, OpenEval follows an equivalent process by converting the input predicate instance into a set of context-based instances from a web search processing, which are given as input to the learned classifier. OpenEval computes the correctness probability of the test predicate instance from the classifier output.

1.1.2 Knowledge Integration and Time-Budgeted Query Answering

In order to determine the response to a new query posed to OpenEval (e.g., *is sugar a healthy food?*), it is useful, in addition to unstructured information on the Web, to integrate

opinions from any other information extraction system. OpenEval aggregates the output of any available Knowledge Resource (KR) system, where each KR system is competent at extracting information from certain types of data (e.g., unstructured text, structured tables on the Web, etc.). These knowledge resources vary significantly in the type of the data and techniques they use, and therefore also vary in their precision, recall, and delay in responding to different types of queries.

The ability to integrate knowledge from different information extraction systems imposes a new challenge - how to automatically decide which system should be queried to provide an accurate answer on demand, given the limited time budget of the system that is making the query. AskWorld, our proposed time-budgeted technique, aims to satisfy such information needs by learning a policy for making specific subqueries to a diverse collection of available knowledge resources, attempting to optimize the query response it can provide within the allotted time budget, and integrating the results returned by the different resources.

1.1.3 Measuring Credibility of Sources and Incorporating the Feedback

This thesis extends OpenEval and AskWorld systems to analyze content of different sources of information on the Web, measure credibility of information sources, and aggregate the collected information in order to make a more informed decision about whether a claim is true or false. We contribute ClaimEval, a novel and integrated approach which given a set of claims to validate, uses OpenEval to extract a set of pro and con arguments from the web information sources, then jointly estimates both the credibility of sources and correctness of claims. ClaimEval uses Probabilistic Soft Logic (PSL), resulting in a flexible and principled framework which makes it easy to state and incorporate different forms of prior knowledge.

ClaimEval also allows us to integrate users' feedback in a principled way and extends applicability of our IE technique. By performing a task in the real world, we will be able to get feedback on how the agent has performed and incorporate the feedback to improve the accuracy of our system. For example, in our mobile robot, if ClaimEval incorrectly infers

the location of object *"coffee"* as *"printer room,"* the robot can assess the correctness of this knowledge by going to the printer room and checking if coffee can be found there. Another source of feedback is the interaction that CoBot has with people who can help it to acquire examples of objects and the corresponding place where the robot can find the object. We will explain how ClaimEval can incorporate these feedbacks in its evolution, in a principled way.

1.1.4 Combining Retrieval and Use of Knowledge

We show how information extraction techniques developed in this thesis can be used to provide knowledge to anytime intelligent agents, in particular for a find-and-deliver task in a *real mobile robot (CoBot)* and for a *trip planner agent*.

For CoBot, we investigate a *find-and-deliver* task, where a person specifies an object in open-ended natural language (e.g., "coffee") and the robot needs to find-and-deliver the object to a specified destination (e.g., "GHC-8125"). This is a challenging problem because robots have limited knowledge and perception, and people use highly variable language when specifying a task. We show that OpenEval enables the robot to actively query the Web to learn new background knowledge about the physical environment. In this context, OpenEval returns the probability that an object can be found in the location queried by the robot. For example, OpenEval learns the probability distribution that indicates a possible location (e.g., "kitchen") to contain an object (e.g., "coffee"). This probability is then dynamically incorporated into a utility function, which takes into account the travel distance to a location, the number of human interactions required to get to a location, and the observation of the object during previous executions at that location. The robot then infers the maximum-utility plan which corresponds to a sequence of locations it should visit, asks a human to provide it with the object, and then takes the object to a destination.

We also show that OpenEval can be used to provide knowledge to a trip planner agent. Currently, planners assume that a problem includes the predicate instances. However for planners to address real-world problem, it is not feasible to enumerate all the instances. For example, a trip planner may require the names of cities, hotels, museums, restaurants, etc, but it will not make sense to input to the planner the complete set of such entities.

The assumption that the knowledge is provided as an input to the planner, limits the applicability of the current planning techniques. We contribute a novel method for a planner to actively query OpenEval to acquire instant knowledge about the planning problem. We introduce a new technique, called OpenWorld, that estimates the relevant knowledge to the initial state and the goal of a planning problem, and then effectively generates corresponding queries to the WWW using our OpenEval query system. We then contribute a planner that iteratively adds knowledge to the planning problem and searches for a solution by calling Fast-Downward planner [Helmert, 2006b].

1.2 Contributions

In this thesis, we make the following contributions:

- OpenEval: Anytime Web Information Query Evaluation
  - A novel and fully automated anytime IE approach that learns to evaluate the correctness of a claim using the unstructured information on the Web.
  - A novel exploration/exploitation search approach, that allows OpenEval to navigate the diversity of information that exists on the Web.
  - Empirical results with our model illustrate effectiveness of our approach compared to related techniques. We show that OpenEval is able to respond to the queries within a limited amount of time while achieving high F1 score.
  - The use of OpenEval in three different domains: robotics, drug discovery, and password-generation programs, and experimental results showing that OpenEval is able to correctly validate the correctness of new predicate instances for each of these domains.
- AskWorld: Knowledge Integration and On-Demand Time-Budgeted Query Answering
  - A new Knowledge-on-Demand (KoD) service that aggregates opinions from multiple knowledge resources to return the most accurate response to a query.

  - A budget-sensitive knowledge integration technique that is designed to provide a best-effort response within the time-budget specified by the user or the application.
  - Extensive experiments on real world datasets demonstrate AskWorld's capability in selecting most informative resources to query within test-time constraints, resulting in improved performance compared to competitive baselines.

- ClaimEval: Measuring Credibility of Sources and Extracting Reasons

  - A fully-integrated technique for measuring credibility of sources and validating the truth of claims.
  - A flexible and principled framework for joint credibility estimation and claim evaluation using Probabilistic Soft Logic (PSL) [Kimmig et al., 2012; Broecheler et al., 2010].
  - Extensive experiments on real-world datasets demonstrate ClaimEval's capability in determining validity of a set of claims, resulting in improved accuracy compared to state-of-the-art approaches.

- ObjectEval: Using the Web to Interactively Learn to Find Objects

  - An approach for learning background knowledge about the environment by querying OpenEval system.
  - An approach for finding and delivering objects that dynamically instantiates a utility function using the results of a Web query, and which interactively learns about the physical environment by getting feedback from humans.
  - A demonstration of our system, enabling CoBot to find and fetch objects in a real- world environment.

- OpenWorld Planner: Iterative Query-Based Open World Planning

  - An automated planning approach that actively queries the Web using OpenEval to add needed knowledge until the planner solves the planning problem.

  - Experiments show our approach is able to decrease the number of queries sent to the Web by a factor of at least two orders of magnitude compared to the baseline approach.

1.3 Evaluation

We evaluate our algorithms in two ways:

- **Datasets:** We test all of our information extraction approaches on a wide range of predicates chosen randomly from sources such as Freebase, NELL, and Wikipedia. We use accuracy, precision, recall, and F1 score as the main metrics, and compare our results to the baseline and the state-of-the-art techniques.
- **Simulated environment:** We test our ObjectEval approach in a realistic simulated environment, where our simulated robot is able efficiently and automatically query ObjectEval to find novel objects. Our simulated environment consists of 290 spaces of an office building, where only a topological map and the space types (e.g., "office," "bathroom" etc.) are known to our system.

We also demonstrate the application of our information extraction techniques on a mobile office robot, that actively queries OpenEval, and successfully finds and delivers objects to different rooms in Gates Hillman Center office building.

1.4 Document Outline

The thesis is organized as follows:

- **Chapter 2** introduces OpenEval and formally defines how *claims* are represented using multi-argument predicate instances. This chapter also explains the experimental results comparing OpenEval to baseline approaches.

- **Chapter 3** discusses how OpenEval can be extended to an on-demand, time-budgeted query answering approach. We also explain how OpenEval can be extended to integrate the output of other knowledge harvesting systems in order to provide more accurate responses. We demonstrate that our technique is capable of selecting the most informative resources to query within test-time constraints, resulting in improved performance compared to our approach presented in Chapter 2.

- **Chapter 4** presents our ClaimEval approach which jointly estimates both the credibility of sources and correctness of claims. We explain the detail of extensive experiments on real-world datasets and compare the accuracy of the ClaimEval approach to state-of-the-art approaches, including OpenEval.

- **Chapter 5** discusses how OpenEval is used in a mobile robot to interactively learn to find objects in an office environment. We present the detail of experiments showing that acquiring knowledge through OpenEval can significantly decrease the number of interactions that robot needs to have with humans, compared to a baseline approach which uses no background knowledge.

- **Chapter 6** presents our automated planning approach that actively queries OpenEval to add needed knowledge. We show that our technique is able to plan a trip for a variety of different problems when no knowledge is given as input to the planner.

- **Chapter 7** discusses areas of related research, including information extraction, robotics, semantic web, and machine learning community, and places this thesis into the context of these works.

- **Chapter 8** summarizes our major findings and suggests potential lines of future research.

Chapter 2

Anytime Approach for Web Information Query Evaluation

In this chapter, we focus on how to determine the value of a given proposition by devising and implementing a new learning approach, called OpenEval. OpenEval evaluates the correctness of propositions that are stated as multi-argument predicate instances using the data on the Web.

We begin in Section 2.1 by motivating the problem of extracting structured knowledge from the Web, and explaining the problem of *information validation*. Section 2.2 presents the details of the OpenEval approach. In this section, we first explain the details of our algorithm that converts a set of seed examples to a set of Context-Based Instances (Section 2.2.1). Our learning algorithm to train a set of classifiers, that are later used to evaluate the correctness of a set of predicate instances, is described in Section 2.2.2. The online algorithm for evaluating the correctness of a test predicate instance is explained in Section 2.2.3. Section 2.3 reports our experimental results. Section 2.4 presents the summary of this chapter.

2.1 Introduction

Arguments for the necessity of encoded knowledge for AI date back to McCarthy (1959). Several subfields of AI such as question answering [Harabagiu and Hickl, 2006; Magnini et al., 2002], information extraction [Etzioni et al., 2004] and robotics [Kollar and Roy, 2009] all profit from machine accessible knowledge. In all these cases, an external application such as a Question Answering (QA) system needs access to a knowledge-base (KB). The usual approach to build a wide-coverage knowledge-base is to use Information Extraction (IE) techniques that extract a large set of facts from corpora such as the Web [Etzioni et al., 2004, 2008; Carlson et al., 2010d]. In this chapter, we address a slightly different problem by assuming that the application interacts with the KB system by requiring the assessment of a claim (i.e., hypothesis), e.g. the agent may ask the KB system to assign the correctness probability to the claim *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)* ['Does aspirin cause gastrointestinal bleeding as a side effect?']. This is a specific sub-problem of knowledge acquisition where our goal is to decide whether a given claim is true or false (or likely to be true/false).

Evaluating the correctness of a proposition is of interest to several applications. For example, it has been shown that the accuracy of a QA system can be improved by using automated answer validation techniques that assess the correctness of the answers found by the system [Harabagiu and Hickl, 2006; Magnini et al., 2002]. In automated Question Answering systems, most of the questions and their candidate answers can be converted to a proposition that requires assessment, e.g., the question "What are the side effects of taking aspirin?" and the candidate answer "gastrointestinal bleeding" can be converted to the proposition "Gastrointestinal bleeding is a side effect of taking aspirin." Furthermore, it is shown that the accuracy of IE systems such as KnowItAll [Etzioni et al., 2004], TextRunner [Etzioni et al., 2008] and NELL [Carlson et al., 2010d; Mitchell et al., 2015a] can be improved by validating the correctness of the extracted facts. Finally, in Chapters 5 and 6 we show that a query evaluation system can help to learn the background knowledge about the environment for collaborative robots [Veloso et al., 2012], learn task-relevant knowledge from human-robot dialog, and retrieve knowledge for a planner.

In this chapter, we contribute a novel technique to respond to predicate-based queries

within a limited amount of time while also achieving high recall (at a small cost of sacrificing precision). We focus on how to determine the value of a given proposition by devising and implementing a new learning approach, OpenEval. OpenEval evaluates the correctness of queries that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect(Aspirin, GI Bleeding)*). A predicate such as *p(*x_1,...,x_n*)* defines a relationship between entities *(*x_1,...,x_n*)*. We call *(*x_1,...,x_n*)* an *instance* of predicate p.

As part of training, OpenEval is provided by a set of predicates (e.g., *Drug(x)*, *DrugHasSideEffect(x,y)*) and a few seed examples with each predicate (e.g., *"aspirin"* and *"acetaminophen"* for predicate *Drug(x)*). For each predicate, OpenEval trains a classifier by taking a small number of instances of the predicate as an input and converting each instance into a Web search query. It then extracts a set of positive and negative Context-Based Instances (CBI) from the webpages returned by the search engine. The extracted CBIs are used as training data. Each CBI is a bag-of-words, extracted from the context around the words in the predicate instance. In principle, a single predicate instance could return as many snippets as are on the Web referencing it. Thus, given a small set of instances of the target predicates, OpenEval is able to extract many training instances from the Web.

The extracted CBIs are then used to train a classifier (e.g., SVM) for each predicate. The training CBIs that are extracted for a predicate are treated as positive examples. There is a separate set of negative training CBIs which are used as negative examples of the predicate and is obtained by sampling a portion of the training instances that are extracted for all other mutually-exclusive predicates. Given the set of positive and negative training CBIs, the goal of the classifier is to evaluate the correctness of a new predicate instance.

To evaluate a new predicate instance, OpenEval follows a similar process by converting the input predicate instance to a search query and extracting a set of context-based test instances from the Web, but then gives the extracted CBIs to the trained classifier. For each of the extracted test CBIs, we say that it *supports* the predicate p, if it is classified as positive by the classifier that is trained for predicate p. The CBI *does not support* the input predicate instance, if it is classified as negative by the classifier. The probability of the input predicate instance is calculated by dividing the number of CBIs that support the predicate, by the total number of extracted CBIs. To navigate the diversity of infor-

mation that exists on the Web, we have presented a novel exploration/exploitation search approach, which enables formulating effective search queries and increases the accuracy of responses provided by OpenEval.

We tested OpenEval on a wide range of predicates chosen randomly from Freebase [Bollacker et al., 2008a], a large semantic database of several thousand categories and relations. The baselines for our comparison are the Pointwise Mutual Information (PMI) technique, [Turney, 2001], and weakly-supervised classification approach [Zhang, 2004]. We show that OpenEval significantly improves the F1 score on the test data compared to the baseline techniques.

Furthermore, to illustrate the applicability of OpenEval we show the use of OpenEval on three different domains. In the first domain, we show that OpenEval can be used in a mobile robot to determine the name of a place where an object is likely to be located in an indoor environment. This has an important application in robotics since robots must be able to autonomously follow commands that involve finding objects. In Chapter 5, we will further explain the detail of this approach and will explain how OpenEval is used in this setting to help a real mobile service robot. In the second domain, we use OpenEval in a password-generation algorithm [Blocki et al., 2013] that needs to check if a given word is "picturable." For instance, words such as "car" and "assumption" are, respectively, examples of picturable and non-picturable words. In the third domain, we test OpenEval on DrugBank, a comprehensive database containing information on drugs and drug targets. The thoroughness of DrugBank is such that its information is relied upon worldwide by medicinal chemists, pharmacists, physicians, students, as well as the general public. Over 7,500-drug entries are available (about 1,500 entries are FDA approved; about 6,000 are experimental). However, users manually curate DrugBank, and, therefore, drug entries are subject to human error. OpenEval is used to validate the information in the DrugBank database related to drug interactions. We will show that OpenEval can be used to automatically find human errors entered in the database. For all these three domains, we perform an evaluation of the predictions that OpenEval makes and show that OpenEval is able to correctly validate most of predicate instances. For the first two domains, we also show that OpenEval can outperform a state-of-the-art technique that is specifically designed for those domains.

Finally, we use OpenEval as part of the Never-Ending Language Learning (NELL) [Mitchell et al., 2015b] project to improve accuracy of information that is extracted by NELL. NELL is a computer program that learns to read the Web 24 hours/day. NELL has been continuously reading the data on the Web since January 2010, and so far has acquired a knowledge base with over 80 million confidence-weighted beliefs, where beliefs are represented by one- or two-argument predicates. Although NELL extracts a diverse set of beliefs, the extracted knowledge is not always correct. To ensure that NELLs extractions are correct, NELL uses OpenEval to evaluate the correctness of its own extractions. The probability that is computed by OpenEval for each of NELL's extractions, is used to decide if a particular predicate instance extracted by NELL should be promoted in order to be shown in the final knowledge base. In the experimental section, we present the detail of experiments showing that OpenEval can significantly improve the accuracy of NELL, in particular by improving its *F1* score from 14.67 to 16.07 in the KBP 2013 Slot Filling task [Bryan Kisiel, 2013].

2.2 OpenEval

OpenEval evaluates the correctness of propositions that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect(Aspirin, GI Bleeding)*). A predicate such as *p(*x_1,...,x_n*)* defines a relationship between entities *(*x_1,...,x_n*)*. We call *(*x_1,...,x_n*)* an *instance* of predicate p where each x_i is an argument of such a predicate instance.

OpenEval includes several components, which we capture in its architecture in Figure 2.1. These components are: Learning, Predicate Instance Evaluator, and Context-Based Instance (CBI) Extractor. The role of the learning component is to build a set of training data, train a set of classifiers, and construct a set of *reference sets*. A reference set consists of a set of keywords, that with the trained classifiers, are later used by the predicate evaluator. The predicate evaluator evaluates the correctness of an input predicate instance and returns a probability. Both learning and predicate instance evaluator components call the CBI extractor, in order to extract a set of CBIs from the Web.

OpenEval considers a set P of predicates p with instantiation i_p for each predicate.

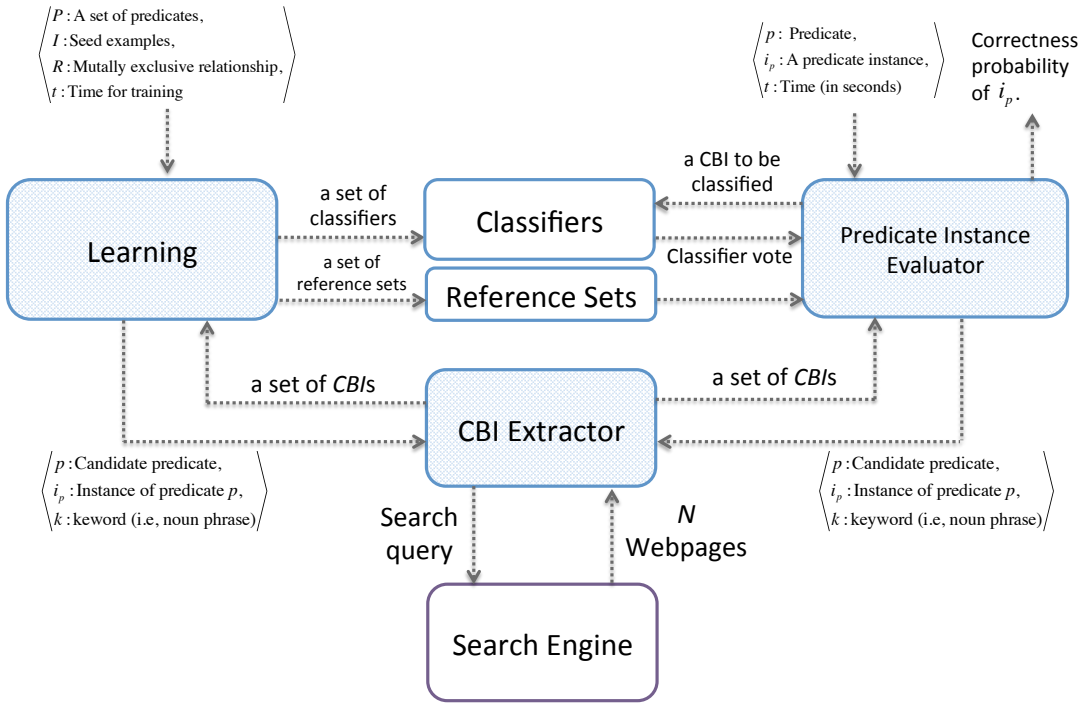


Figure 2.1: The architecture of OpenEval. OpenEval consists of three main components: Learning, Predicate Instance Evaluator, and Context-Based Instance (CBI) Extractor.

For example, for $p = \textit{Drug}$, values of i_p are *Aspirin* and *Acetaminophen*, and for $p = \textit{DrugHasSideEffect}$, values of i_p are (*Aspirin, GI Bleeding*) and (*Acetaminophen, Nausea)*. OpenEval evaluates the correctness of a combined triple $t = \langle p, i_p, t \rangle$, where t is the input time that OpenEval is allowed to use for evaluation.

In the next section, we describe in detail the three main components of OpenEval, which we capture in its architecture in Figure 2.1, namely: Context-Based Instances (CBI) extractor (Section 2.2.1), Learning (Section 2.2.2), and Predicate Instance Evaluator (Section 2.2.3).

2.2.1 Context-Based Instance (CBI) Extractor

Algorithm 1 shows the procedure to extract a set of Context-Based Instances (CBIs). Each CBI consists of a set of features constructed by querying the open Web and processing the retrieved unstructured webpages. The input to the *CBI Extractor* is the tuple $\langle p, i_p, k \rangle$, where p is the input predicate (e.g., *DrugHasSideEffect*), i_p is an instance of predicate p (e.g., *(Aspirin, GI Bleeding)*), and k is a keyword that is used to formulate the search query (e.g., *side effects*). The output of the CBI extractor is a set of context-based instances which are extracted for the input instance i_p.

Algorithm 1 OpenEval - Context-Based Instance (CBI) Extractor

Require: $\langle p, i_p, k \rangle$ //p: predicate, i_p: predicate instance, k: keyword
1: **Function: CBIExtractor** ($\langle p, i_p, k \rangle$)
2: $B_{i_p} \leftarrow \phi$ //B_{i_p} is the set of all the CBIs for i_p
3: $pArgs \leftarrow$ Arguments of i_p separated by space
4: $Q \leftarrow$ "$pArgs$ k" //Q is the search query
5: $W \leftarrow$ Retrieve the first N documents for query Q
6: $B_{W_i} \leftarrow \phi$ //B_{W_i} is the list of all the bags-of-words that are extracted from webpage W_i
7: **for all** webpages $W_i \in W$ **do**
8: **for all** occurrences of words in *pArgs* (close to each other) in W_i **do**
9: $\bar{t} \leftarrow$ extract text around *pArgs*,
10: Remove stop words and words in *pArgs* from $\bar{t}$
11: Add $\bar{t}$ as a bag-of-words to B_{W_i}
12: **end for**
13: Add bags-of-words in B_{W_i} to B_{i_p}
14: **end for**
15: **return** B_{i_p}

Given the input tuple, the CBI Extractor first builds the search query (Lines 3-4). The search query Q is built from arguments of the input instance i_p, and input keyword k. For example, query $Q =$ {*"Aspirin" "GI Bleeding" side effects*} is built for the predicate instance *DrugHasSideEffect(Aspirin, GI Bleeding)*, where the keyword is *side effects*. The CBI extractor then searches the query Q in Google and downloads the first N webpages (Line 5). For each webpage W_i, which the search engine finds, the CBI extractor searches

the content of W_i and finds all the positions in W_i whose words in i_p appear "close" to each other (Line 8). "Close" means that we allow words in i_p to appear in any arbitrary order and up to a maximum of 15 words can be in between, before and after them. For each occurrence of words in i_p, consider $\bar{t}$ as the text that occurs around words in i_p (Line 9). All the stop words and the words in i_p are deleted from $\bar{t}$ and the remaining words in $\bar{t}$ are then added as a bag-of-words into the set B_{W_i} (Line 11). Finally all the bags-of-words that are extracted from different webpages are returned as a set of extracted CBIs for input instance i_p (Line 13).

To show an example of Context-Based Instances (CBIs) that are extracted by the CBI extractor, consider the text in Figure 2.2 as part of a text of a webpage that is retrieved for predicate instance *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*.

Common Aspirin Side Effects: The common **aspirin** side effects are primarily related to its anti-platelet effects. **Gastrointestinal bleeding**, is the most common serious side effect. Aspirin also frequently causes gastrointestinal irritation including gastritis, gastric and duodenal ulcers, and esophagitis. These combined with the anti-platelet effect which can cause any bleeding to be more severe make GI bleeding a common **aspirin** side effect. **Aspirin** when used in combination with other NSAIDs can greatly increase the risk of **gastrointestinal bleeding**.

Figure 2.2: An example of the text that is found for predicate *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*.

In the example that is demonstrated in Figure 2.2, the arguments of i_p are "aspirin" and "gastrointestinal bleeding", and therefore the value of $pArgs$ in Algorithm 1 is *"aspirin" "gastrointestinal bleeding"*. There are only two places where a maximum of 15 words occur between words in $pArgs$. For each of these two cases, we have shown the arguments of the predicate *DrugHasSideEffect* in bold. The underlined text shows the value of $\bar{t}$ for each case (i.e., text around arguments of i_p). The CBI Extractor returns two bags-of-words: {*Common, Common, side, effects, common,...*} and {*severe, make, GI, bleeding,...*}. Each

of these bags-of-words will be used as a separate CBI in B_{i_p}. The algorithm returns a set of CBIs (i.e., B_{i_p}), represented by a set of bag-of-words.

Detecting noisy blocks: The text around the arguments of a predicate can potentially contain non-informative content, e.g. ads or spam. This may cause the noise in the extracted context-based instances. One of the main reasons for noisy extraction is that a webpage typically contains many information blocks, such as navigation, copyright, privacy notice, advertisements, etc., which we call *noisy blocks*. Figure 2.3 shows one of the webpages that are extracted for the query *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*, where the main information block is shown by a solid line, and any other information in the website is considered as noisy blocks. It has been shown by the researchers [Lin and Ho, 2002; Yi et al., 2003] that noisy blocks can significantly harm the accuracy of Web information extraction systems, since they usually have relevant keywords but irrelevant content. Therefore, detecting and removing the noisy blocks can significantly improve the accuracy of our IE technique.

In addition to detecting the noisy blocks, extracting the content of the *main block* is also a challenging problem. Figure 2.4 shows the Document Object Model (DOM) representation of the content of a webpage that contains a snippet related to the predicate instance *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*. DOM is an application programming interface (API) that represents valid HTML and XML documents in the tree structure format. The DOM tree provides a wide range of flexibility for representing and parsing the structure of a single webpage. In DOM tree representation, tags are internal nodes and the text of snippets, links to images, or hyperlinks are the leaf nodes.

The Figure 2.4 shows that extracting the content of a snippet such as *"Common Aspirin Side Effects: The common aspirin side effects are primarily related to its anti-platelet effects. Gastrointestinal bleeding [...]"* is non-trivial, due to the special HTML-formatting that is used to represent the content of the snippet in the HTML page. In the figure, this snippet is shown as one of the children of the annotated node with tag **P**. The word "aspirin" in the snippet (one of the arguments of the input predicate instance) is annotated with an anchor tag (i.e., node with tag "**a**"), and therefore is separated from other children of the node **P**. In order for an IE algorithm to extract the complete text of the snippet, it should be able to automatically determine that the node below tag node **a** should be

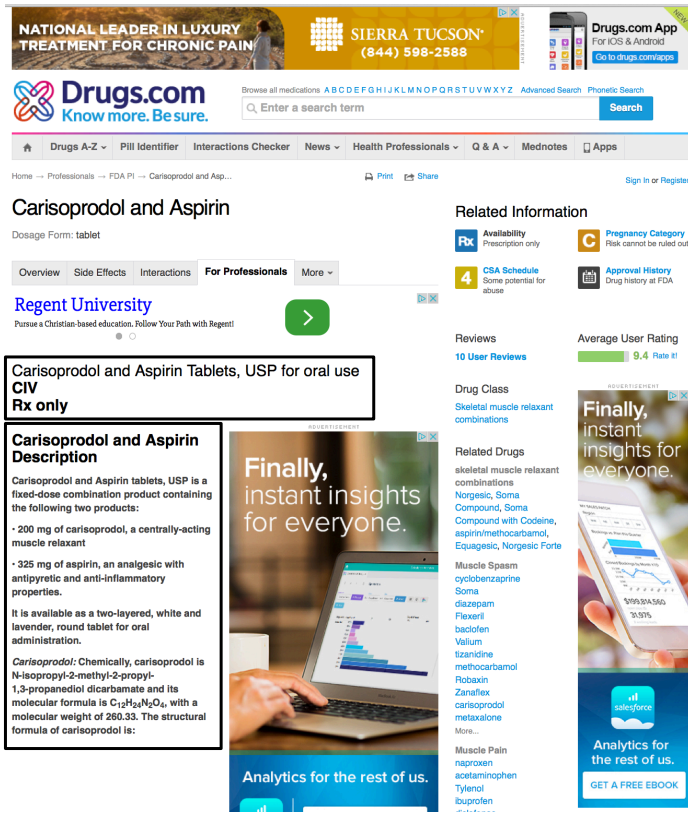


Figure 2.3: An example of a webpage that is retrieved for the query *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*. The solid line is drawn manually and shows the main information block of the webpage. Other information blocks in the webpage are non-informative and should be mostly removed during the extraction.

merged with other children of its parent (node **P**). Determining these cases is non-trivial, as the structure of the DOM tree can be complex and sometimes non-informative blocks (e.g., ads) can be located as one of the children of annotated node **P**.

Although some interesting research has been done on detecting and removing the noisy blocks [Yi et al., 2003; Lin and Ho, 2002; Song et al., 2004; Weninger et al., 2010], they are not fully applicable to our setting. For example, Lin and Ho [Lin and Ho, 2002] proposed an approach to detect the information blocks in news related webpages. However, their work is limited by the assumption that *a prior* knowledge is provided which defines how

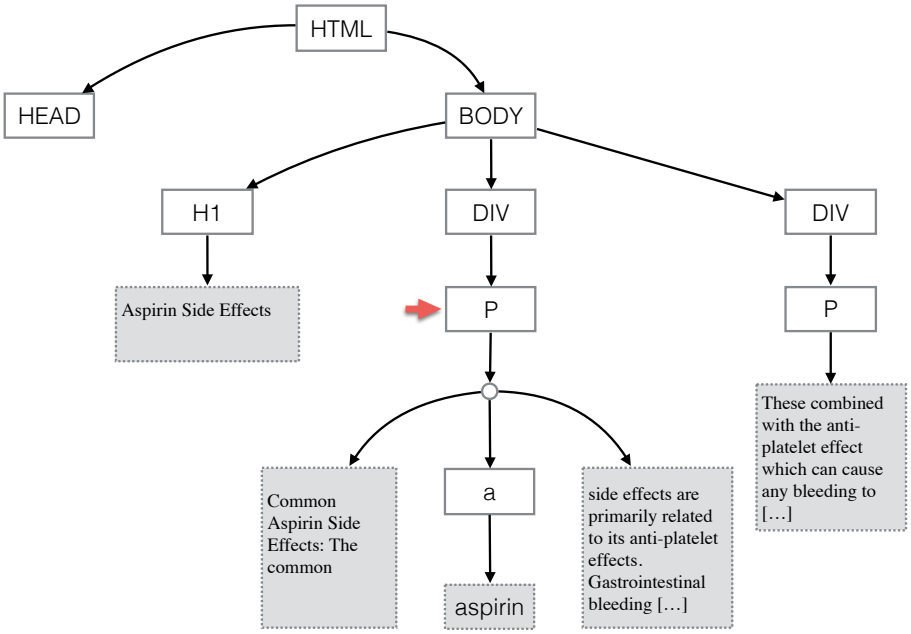


Figure 2.4: A DOM tree representation of part of a webpage that is retrieved for the query *DrugHasSideEffect(Aspirin, Gastrointestinal Bleeding)*. For the snippet *"Common Aspirin Side Effects: The common aspirin side effects are..."*, the text of the snippet is split between different DOM nodes in the tree, e.g. the word "aspirin" is separated from other parts of the snippet since it is represented by the anchor tag node **a**.

a given webpage can be partitioned into different content blocks. This assumption makes their work inapplicable to the setting of OpenEval, since OpenEval retrieves any webpage that is returned by the search engine. It is not feasible to assume that a prior knowledge is available about the the structure of all the returned webpages. Other works, such as [Lin and Ho, 2002], do not require prior knowledge, but instead assume that many of the webpages are retrieved from the same website. The noisy blocks are detected by finding the common layouts or presentation styles between different webpages of a website. Unfortunately, these works are not applicable to OpenEval's setting since it is unlikely for a search engine to return different webpages of the same website.

To address the above challenges, we propose a simple algorithm, which is based on the layout of the HTML, and uses a set of general rules (heuristics) in order to detect most

of the noisy blocks in a webpage. In our experiments, we have observed that our simple extraction algorithm is able to detect most of the noisy content blocks in the webpages retrieved by OpenEval, and further sophisticated algorithms do not necessarily result in achieving higher accuracy. Since detecting the noisy content is not the main focus and contribution of this thesis, we leave the further improvements to our core algorithm as part of the future work. We now give an overview of the DOM (Document Object Model) [1] tree, which is used to represent the structure of the HTML document. We then explain the details of our algorithm.

Each HTML document can be represented by a DOM tree structure, where tags are the internal nodes, and the main content (e.g., text, images, urls, etc.) of the HTML document are represented in the leaf nodes. Figure 2.4 shows an example of a DOM tree node, where the tag nodes are represented by white rectangles, and content nodes are represented by the gray-colored rectangles. Any given HTML document can be represented by a corresponding DOM tree, where the node with tag **root** is the root of the tree. All the viewable elements in a HTML document are shown under the node with tag **BODY**. Algorithm 2 shows our algorithm for extracting snippets of the main content of a webpage.

Algorithm 2 first converts the input HTML document H to a DOM tree (Line 2). It then parses the DOM tree D in depth-first order. For each node n in the tree that it visits, the algorithm calls two functions. The first time that node n is visited, the algorithm calls function *ParseHead*, which takes a node and the depth of the tree as an input (Line 6), and returns *false* if the node contains a non-informative block. For each node of the DOM tree, the function *ParseHead* is called only once. If *ParseHead* detects that the subtree of a particular node contains noisy information, the algorithm then does not parse the content of the node. Next, we explain the details of the *ParseHead* function, shown in Algorithm 3.

Algorithm 3 shows the detail of the *ParseHead* function used by the *Traverse* function, which (i) detects noisy information blocks, and (ii) extracts/merges the texts of the snippets in the subtree of the input node n. From our experiments, we have observed that a few simple heuristics can be extremely helpful in determining most of the noisy blocks. Lines 3-5 shows the heuristics that we have used, which can be summarized as follows:

[1] http://www.w3.org/DOM/

Algorithm 2 OpenEval - Traversing and processing information of the DOM tree in the depth-first order

Require: HTML Document H

```
1:  Snippets ← {}  //Set of all extracted snippets. Public variable that is used by both Traverse and and ParseHead functions.
2:  Function: Traverse (H)
3:  D ← Convert H to a DOM tree
4:  Node node = D.root
5:  depth ← 0
6:  while node ≠ null do
7:    containsNoisyBlock ← ParseHead(node, depth)
8:    if node.childNodeSize() > 0 & containsNoisyBlock == false then
9:      node=node.childNode(0)
10:   else
11:     while node.nextSibling() == null & depth > 0 do
12:       node=node.parentNode()
13:       depth ← depth −1
14:     end while
15:     if node == root then
16:       break
17:     end if
18:     node=node.nextSibling()
19:   end if
20: end while
21: return Snippets
```

- If a DOM node name contains the word *code*, then it is likely to be a code block, and avoid parsing it.
- If the HTML style of a DOM node is set to *display:none*, then do not parse the content of the node, since it is a hidden node and its content is not displayed to the user.
- If the name of a DOM node contains either of the words *ad* or *img*, do not parse the content as it is very likely to contain an advertisement or an image.

Although an adversary can break the above heuristics easily, most of the webpages on the

Algorithm 3 OpenEval - Parsing a DOM node

Require: DOM node n, *depth*

1: *currentSnippet* $\leftarrow$ *""* //Contains accumulated snippet text
2: **Function: ParseHead** (n,*depth*)
3: **if** (*n.nodename == "code"* or *n.attribute("style") == "display:none"*) OR (*n.nodename* contains either *"ad"* or *"img"*) **then**
4: **return false**
5: **end if**
6: **if** n is a text node **then**
7: Clean the text of node n, and add it to *currentSnippet*
8: **else if** n is an Element node **then**
9: **if** (n is a *block* node or has tag name *br*) AND *length(currentSnippet)* > 0 **then**
10: Add *currentSnippet* to *Snippets*
11: *currentSnippet* $\leftarrow$ ""
12: **end if**
13: **end if**
14: **return true**

Web use the standard naming format in the HTML structure, and the current heuristics should be adequate. Moreover, in the next sections we will explain that the classifier that we train for each classifier can also learn to decrease the weight of CBIs that are not informative.

As we are parsing the nodes in the DOM tree, Algorithm 3 accumulates the text of snippets that it sees in the DOM tree, until it reaches either a *block* node or a node with the tag name *br* (Line 9). These two types of nodes determine when we have reached the end of the snippet text. When the algorithm reaches either of these types of nodes, it then adds the content of the accumulated snippet (variable *currentSnippet*) to the list of all extracted snippets (shown by global variable *Snippets* in Algorithm 2). After parsing all the nodes in the DOM tree and extracting the snippets, Algorithm 2 returns the list of all the extracted snippets.

2.2.2 Learning

As part of training, OpenEval is given the tuple $\langle P, I, R, t \rangle$ where P is a set of predicates, I is a set of seed examples for each predicate in P, R is the mutually-exclusive relationships between predicates P, and t is the time that the learning algorithm should spend for the training. To decide if an extracted set of context-based instances belongs to a certain predicate, we train a separate classifier (such as SVM) for each predicate. The OpenEval learning algorithm is shown in Algorithm 4.

Algorithm 4 OpenEval - Learning classifiers

Require: $\langle P, I, R, t \rangle$ //P: a set of predicates, I: seed examples, R: mutual-exclusive relationships of predicates in P,t: time for training (seconds)

1: **Function: LearningClassifiers** ($\langle P, I, R, t \rangle$)
2: $CBI_p \leftarrow$ Call **CBIExtractor** to extract CBIs for all instances of p that exists in I (no keyword is used)
3: $CBI'_p \leftarrow$ Randomly sample and remove a set of CBIs from CBI_p //used to estimate how OpenEval performs on future test data
4: **for all** $p \in P$ **do**
5: $pos \leftarrow CBI_p$
6: $neg \leftarrow$ Sample instances from CBI_i $\forall$ predicates i that are mutually-exclusive to p
7: $C_p \leftarrow$ Train SVM using *pos* and *neg*
8: **end for**
9: **while** elapsed time $\leq$ t **do**
10: $C_{p'} \leftarrow$ Find classifier $C_{p'}$ that has maximum entropy on CBI'
11: $ref \leftarrow$ Extract reference set from $C_{p'}$
12: $k \leftarrow$ Choose a keyword from *ref*
13: $NewCBIs \leftarrow$Extract CBIs for predicate p' using instances I and keyword k
14: $CBI_{p'} \leftarrow CBI_{p'} \cup NewCBIs$
15: Retrain classifier for predicate p'
16: **end while**
17: **return** C, reference sets for all the predicates

The learning algorithm first iterates over all the predicates $p \in P$ and extracts a set of CBIs (denoted by CBI_p) for each predicate p using the input seed examples (Line 2). It also constructs another set, CBI'_p, by randomly choosing and removing 10% of the CBIs from set CBI_p (Line 3). CBI'_p is used as an *evaluation* set during the learning to estimate

how OpenEval performs on the future test data. A SVM is then trained for each predicate p by considering the CBIs that are extracted for p as positive examples (Line 5). To build the set of negative examples, we sample from the instances that are extracted for all other predicates that are mutually exclusive from p (Line 6). Each of these CBI examples is then transformed into a *feature vector*, where each element corresponds to the frequency of a distinct word in the CBI. The dimension of the vector is equal to the total number of distinct words that occur in the training data. We train a classifier for each predicate p so that it classifies the input feature vector f as positive if f belongs to p, and classifies it as negative otherwise (Line 7). The classifier that is trained for p is later used in the evaluation to decide if a new CBI belongs to predicate p.

Iteratively Minimizing Entropy of Trained Classifiers

After training a classifier for each predicate using extracted CBIs (Lines 3-8), OpenEval iteratively chooses a classifier with the lowest accuracy and tries to improve its accuracy by extracting more training examples from the Web. To achieve this goal, our learning algorithm iteratively minimizes the entropy for the trained classifiers (Lines 9-16). We explain the details of our learning algorithm below.

Consider predicate p and assume that CBI is the set of CBIs extracted for predicate p and CBI^* as the set of CBIs extracted for predicates that are mutually-exclusive to predicate p. Also assume:

- C_p: is the classifier trained for predicate p
- CBI^+: A subset of CBIs in CBI that are classified as positive by classifier C_p
- CBI^-: A subset of CBIs in CBI that are classified as negative by classifier C_p
- CBI^{*+} and CBI^{*-} are defined similarly to CBI^+ and CBI^-, but for the CBIs that are extracted from predicates mutually-exclusive to p.
- $C_p(CBI)$: sum of confidence values assigned by C_p to instances in CBI.

Ideally, our learning algorithm should train a classifier that has the *minimum entropy* value for each predicate. Having the minimum entropy value means that the classifier C_p

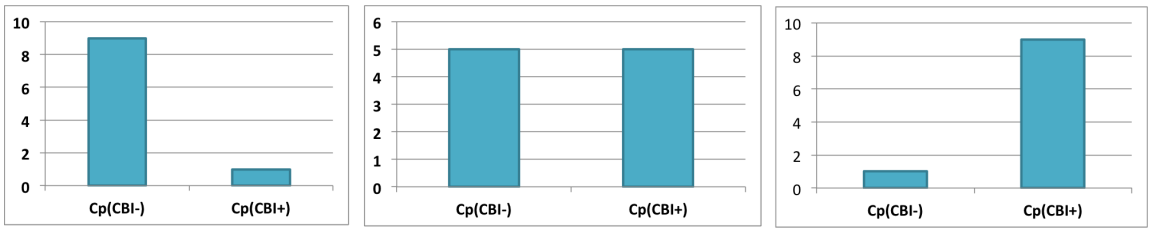
should classify all the CBIs that are extracted for predicate p as positive, and all other CBIs that are extracted for other mutually-exclusive predicates as negative. It should also assign a high confidence value to its prediction. Therefore, the ideal classifier should maximize CBI^+ and minimize CBI^-. Using the entropy notation, this objective can be written as minimizing:

$$E = C_p(CBI^-) * log(C_p(CBI^-)) - C_p(CBI^+) * \log(C_p(CBI^+)) \tag{2.1}$$

Similarly we can write it for set CBI^*:

$$E^* = C_p(CBI^{*+}) * \log(C_p(CBI^{*+})) - C_p(CBI^{*-}) * \log(C_p(CBI^{*-})) \tag{2.2}$$

Our learning algorithm minimizes the value of $E + E^*$. Figure 2.5 shows an example of different values of E, for different values of $C_p(CBI^+)$ and $C_p(CBI^-)$.



E=8.5 E=0 E=-8.5

Figure 2.5: Entropy values (value of E in Equation 2.1) for three different cases. The first figure from the left shows that the entropy value is high since the classifier has classified most of the (positive) instances in the set CBI as negative. The figure in the middle shows that the entropy value is equal to zero (the classifier is performing almost randomly). The figure on the right shows that the entropy value is low since the classifier is able to classify most of the instances correctly (classifying instances in CBI as positive).

To minimize the entropy value of classifiers, the learning algorithm iteratively finds a classifier $C_{p'}$ that has the maximum entropy value on the evaluation set ***CBI'*** (Line 10).

The training data of the predicate which has the maximum entropy is then increased by extracting new CBIs (Lines 10-15). By increasing the number of training data and retraining the classifier, OpenEval tries to decrease the entropy value of a classifier and make it more accurate and confident in its prediction.

To extract new CBIs for a predicate, the learning algorithm first chooses a keyword from the *reference set* that is constructed for the predicate. A reference set contains a set of keywords that represent some of the underlying semantic concepts of a predicate and mostly distinguishes a semantic meaning of a predicate from others. For example, keywords such as {*drug, drug effects, adverse effects*} can be used as part of the reference set for predicate *DrugHasSideEffect(x,y)*. The reference set can be built by selecting a set of relevant phrases from CBIs that are already extracted for a predicate. Different feature extraction techniques that have been studied in the machine learning community can be used to construct the reference set [Blum and Langley, 1997]. Among these techniques, we use feature ranking using weights from a linear SVM classifier which has been shown to be an effective approach for text classification tasks [Mladenić et al., 2004]. Thus, the reference set is constructed by selecting the top $K\%$ keywords that have the highest absolute weights in the trained SVM (i.e., the normal to the hyperplane that separates positive and negative classes).

What does OpenEval learn?

In linguistics, the term "word sense" is used when a word has different meanings. For example, the word "apple" has at least two senses, "apple" as a fruit and "apple" as the name of a company, means that "apple" is an instance of two different predicates: *fruit* and *company*. In this paper we assume that a set of entities $(x_1, ..., x_n)$ can potentially be an instance of different predicates. For example, *(J.C. Penneys, United States)* is an instance of both *PersonBornInCountry* and *CompanyLocatedInCountry* predicates. In this case we say that *(J.C. Penneys, United States)* has two different senses.

By iteratively choosing keywords from a reference set, OpenEval automatically learns how to map a set of entities to an appropriate predicate (i.e., sense) to which they belong. For example, given a predicate instance such as *Fruit(Apple)*, OpenEval uses keywords that are chosen from the reference set of predicate *Fruit* as part of the search query which biases the search engine to return the documents that contain information about *apple* as a

fruit. For example, by using a keyword such as $\{$*vitamin*$\}$, the search query for predicate *Fruit(Apple)* would be $\{$*"Apple" vitamin*$\}$ which forces the search engine to return the results that are more likely to mention "apple" as a fruit rather than as a company. Using a keyword as part of the search query helps OpenEval to automatically extract CBIs that specify an appropriate sense of the input predicate instance. It is worth mentioning that in the first iteration of the learning algorithm (Lines 2-8), no keyword is used to extract CBIs. However, since CBIs are extracted for *a set* of seed examples (not only one), OpenEval eventually would be able to converge to extract relevant keywords for the reference set.

Choosing Negative Examples

To train a classifier for predicate p, a subset of context-based instances that are extracted for the other predicates (mutually-exclusive to predicate p) are randomly chosen as the negative examples of predicate p. The set of negative instances can help to circumscribe the desired keywords for predicate p. For example, negative CBIs that are extracted for other predicates will help the classifier to eliminate keywords that are common across different predicates.

facts, love, :, arranged, forum, kind, article, forum, kind, article, economy, over-standing, followers, rudolph, mandaean, encyclopedia, educational, discussion, ?x, spiritual
facts, arranged, followers, over-standing, educational, kind, opportunities, discussion, article, forum, religion, religion-facts, love, basics, influential, mandaean, spiritual, beliefs, god, faith,

Figure 2.6: The top 20 highest-ranked features that are extracted for predicate *Religion(x)*. The first row shows the features that are learned by the classifier using only two predicates *Religion(x)* and *AcademicField(x)* for training. The second row shows the learned features when all the predicates in Tables 2.1 and 2.2 are used in training. In general, having more predicates in the ontology would improve the accuracy of OpenEval, since a more diverse set of negative examples would be chosen from the ontology.

For example, Figure 2.6 shows a set of words that is learned by the classifier for predicate *Religion(x)* when the words are sorted by their weights in the resulting SVM classifier. The first row shows the result when we have used only two predicates *Religion(x)* and *AcademicField(x)* in the training. The second row shows the result when all the predicates in the ontology (shown in Tables 2.1 2.2 in the experimental result section) have been used in the training. It can be seen that the features in the second row are relatively more informative compared to the first row.

As shown in Figure 2.6, selecting negative examples from a more diverse set of predicates helps the classifier to learn more informed weights for the features, and potentially achieve higher accuracy. For example, assume that an agent (e.g., a semantic parser that needs to annotate a sentence) asks OpenEval to validate if *Himalaya* is a name of a color. The true answer is *false*, since *Himalaya* is the name of a mountain and the predicate *Mountain* is mutually-exclusive to predicate *Color*. Now assume that predicate *Mountain* is not defined in the ontology of OpenEval. In this case, most of the CBIs that are extracted for instance *Himalaya* contains features and keywords that the classifier has not observed during the training, or has not had enough data to learn useful weights. This can cause significant noise in the classification and affect the accuracy. However, if *Mountain* is defined in OpenEval's ontology, then the classifier that is trained for predicate *Color* has learned to classify some of the snippets that are extracted for predicate *Mountain* as negative. Hence, the classifier is more likely to reject *Himalaya* as an instance of predicate *Color*.

The portion of the negative examples and how they have been chosen have an important impact on the accuracy of OpenEval. For example, it has been widely shown in the machine learning literature [Kubat et al., 1997] that the accuracy of learning systems decreases when the negative examples heavily outnumber the positive examples. In our current approach, we take a random sample from all the training instances that are extracted for other mutually-exclusive predicates. In order to bound the size of the negative examples, for each predicate we take 10 times as many negative instances as positive instances. We defer further consideration of other relevant sampling techniques for future work.

Seed Examples

One of the main advantages of OpenEval is that it only needs a few seed examples for training, mainly because the actual training data are created by converting the input seed examples to a set of CBIs. However, similar to any other machine learning techniques, providing more seed examples helps OpenEval to achieve higher accuracy. For example, consider predicate *AcademicField(x)*. Figure 2.7 shows the top 20 highest-ranked features that are learned for the predicate *AcademicField(x)* when we train OpenEval using different number of seed examples. As expected, it can be seen that the features are more informative as we increase the number of input seed examples to OpenEval.

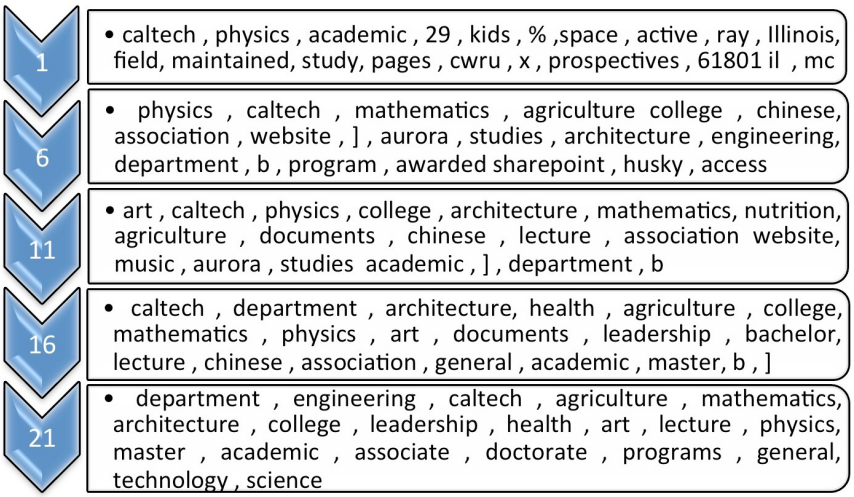


Figure 2.7: The top 20 highest-ranked features that are learned for predicate *AcademicField(x)*. The number of seed examples that are used in the training is shown on the left.

2.2.3 Predicate Instance Evaluator

To evaluate a new predicate instance, OpenEval follows a similar process to the learning algorithm by converting the input predicate instance to a set of CBIs, but gives the extracted CBIs to the trained classifier to compute the correctness probability of the input predicate instance. The input of the predicate evaluator is predicate p, the candidate instance i_p, and time t (seconds). OpenEval outputs the probability of i_p to be an instance of

predicate p.

To evaluate if i_p is an instance of predicate p, OpenEval iteratively selects a set of keywords from the reference set that is built for predicate p (the reference set is built during the training) and calls the CBI Extractor to extract a set of CBIs for i_p using the selected keywords. The goal is to select a set of keywords and to formulate search queries that capture different senses/aspects of a predicate. For example, predicate *UniversityProfessor(x)* is related to different aspects such as *teaching professor* and *research professor*. Different aspects of this predicate can be captured by keywords such as {*research, teaching, paper,* ...}. To decide which keywords should be chosen, we define a utility function that measures the utility value of each individual keyword using the documents that are retrieved from the Web. Given this utility function, we propose a learning algorithm that iteratively explores/exploits different keywords from the reference set, retrieves documents from the Web, and extracts CBIs from the retrieved documents.

Let *ref* $= \{k_1, \ldots, k_n\}$ be a reference set that is extracted for predicate p, where each k_i is one of the keywords that is extracted during the training. Also consider set K as a subset of keywords that are selected from *ref* and CBI_K as a set of CBIs that are extracted for input predicate instance i_p using keywords in K. To construct CBI_K, we go through all the keywords in K and extract a set of CBIs for i_p using the selected keyword. We express the utility of the set of keywords K as the following:

$$U(K) = | \sum_{c_i \in CBI_K^+} conf(c_i) - \sum_{c_i \in CBI_K^-} conf(c_i) | \tag{2.3}$$

where CBI_K^+ is a set of all CBIs that are classified as positive by the classifier of predicate p and CBI_K^- is a set of all CBIs that are classified as negative. $conf(c_i)$ is the confidence value that is assigned to c_i by the classifier. The utility $U(K)$ shows how keywords in K could help to classify i_p either as positive (true) or negative (false) classes. If predicate instance i_p belongs to predicate p, then we would like to find keywords K to extract CBIs that give the highest confidence value on classifying i_p as positive (large value of $U(K)$). On the other hand, if i_p does not belong to predicate p, then we would like to find keywords that help to classify i_p as negative where they give the highest confidence on the classification vote. Therefore, our goal is find a set of keywords K that maximizes the

utility value $U(K)$:

$$\arg\max_{K} U(K) \tag{2.4}$$

To maximize the expected utility, one naive approach is to rank each of the keywords based on its utility and always choose the keyword that has the highest utility value. We call it a greedy approach. This approach may deprive us of the opportunity of finding the optimal set of keywords K. Therefore, we face a trade-off between exploitation and exploration to gain the optimal overall utility [Gittins and Jones, 1974]. To achieve both exploitation and exploration goals, we rank the keywords based on their expected utility plus a term related to their variances, instead of solely using the expected utility as in the greedy approach. The term related to the variance is known as the exploration bonus [Sutton, 1990]. We use a similar algorithm to the UCB1 algorithm [Auer, 2003] which aims at obtaining a fair balance between exploration and exploitation in a K-Armed bandit problem, in which the player is given the option of selecting one of the K arms of a slot machine (i.e., the bandit). In the context of our work, each keyword in the reference list *ref* is treated as an arm which can be used to formulate a search query and extract the correspondence CBIs. The selection of keywords is directly proportional to the number of times that all the keywords have been selected and inversely proportional to the number of times that each keyword is selected. We also assume that an initial reward value is assigned to each keyword which is calculated from the initial weight assigned to each keyword during the training. More precisely, we assign $Q(k)$ value to each keyword k:

$$Q(k) = \bar{U}_k + C' \sqrt{\frac{2 \log (n+1)}{N_k + 1}} \tag{2.5}$$

The keyword k that maximizes $Q(k)$ is selected at each iteration. $\bar{U}$ is the average utility (reward) that we have obtained by selecting keyword k after n iterations, N_k is the number of times that keyword k is selected. C' controls the balance between exploration and exploitation.

Algorithm 5 shows the detail of our technique for evaluating the correctness of an input predicate instance i_p. We first extract the reference set for the predicate p (Line 3).

The average reward for each keyword is initialized by the weight that is assigned to it in the reference set divided by the sum of all the weights of keywords in the reference set (Line 5). The algorithm then iteratively updates Q values for all the keywords (Line 8) and extracts CBIs using the keyword that has the maximum Q value (Lines 9-10). The utility for the selected keyword is then calculated (Line 12) and the average utility value ($\bar{U}_{k'}$) is updated (Lines 13-15). Finally, the algorithm classifies each of the extracted CBIs (Lines 17-18). The next step of the algorithm is to combine the classification results for all the extracted CBIs.

Algorithm 5 OpenEval - Predicate instance evaluator

Require: $\langle i_p, C, t \rangle$ //i_p: a predicate instance to be evaluated, C: classifiers trained for predicates, t: time (seconds)

1: **Function: Evaluator** ($\langle i_p, C, t \rangle$)
2: *CBI* $\leftarrow \{\}$ //set of CBIs for predicate p
3: *ref* $\leftarrow$ Extract reference set from classifier C_p
4: $N_k \leftarrow 0 \ \forall k \in$ *ref* //number of times that k is used as a keyword
5: $\bar{R}_k \leftarrow \frac{\text{weight of } k \text{ in } ref}{\text{sum of weights in ref}} \quad \forall k \in$ *ref* //initializing average reward obtained by using keyword k
6: $n \leftarrow 1$
7: **while** elapsed time $\leq$ t **do**
8: $Q_k \leftarrow Q(\bar{R}_k, N_k, n) \ \forall k \in$ *ref* //update Q values
9: $k' \leftarrow$ Choose keyword with maximum Q value
10: $B \leftarrow$ **CBIExtractor**($\langle p, i_p, k' \rangle$)
11: *CBI* $\leftarrow$ *CBI* $\cup B$
12: $U \leftarrow \sum_{[c_i \in B^+]} conf(c_i) - \sum_{[c_i \in B^-]} conf(c_i)$
13: $N_{k'} \leftarrow N_{k'} + 1$
14: $\bar{U}_{k'} \leftarrow (\bar{U}_{k'} + |U|)/N_{k'}$
15: $n \leftarrow n + 1$
16: **end while**
17: Classify instances in *CBI*
18: $eval \leftarrow$ calculate weighted majority vote for instances in *CBI*
19: **return** true *if* $eval \geq 0.5$; false *otherwise*

Combining the Classification Results

The simplest way to combine the classification results from multiple CBIs is within a voting framework. In this framework, the correctness probability of the input predicate

instance i_p is calculated by dividing the number of instances that are classified as positive by the total number of instances that are returned by the CBI Extractor, where the classifier vote is weighted by its confidence value. This scheme is known as weighted majority (or plurality) voting which has been shown to be very robust compared to other more intelligent voting schemes that have been used in forecasting literature [Clemen, 1989; Dietterich, 1998]. The correctness probability of i_p can be written as follows:

$$P(i_p = t|\text{CBIs}(i_p)) = \frac{1}{Z} \sum_{c \in \textit{CBIs}(i_p)} W_{[C_p(c)=t]} \times 1\{C_p(c) = t\} \tag{2.6}$$

where Z is the normalization factor:

$$Z = \sum_{b=\{t,f\}} \sum_{c \in \textit{CBIs}(i_p)} W_{[C_p(c)=b]} \times 1\{C_p(c) = b\} \tag{2.7}$$

and $W_{[C_p(c)=t]}$ is defined as the confidence of classifier C_p where it classifies c as positive.

2.3 Experimental Evaluation

We designed our experiments to study the following questions.

1. What is the accuracy of OpenEval in terms of recall and precision and how does it compare to the PMI technique [Turney, 2001] and weakly-supervised relation classification [Zhang, 2004] for different types of predicates?

2. How is the accuracy of responses provided by OpenEval affected if we use different parameter settings? In particular, how is the accuracy affected when more time is given for training and evaluation?

3. How does OpenEval perform when it is used in different domains or applications? In particular, how does it perform in three different domains: *finding objects in an environment*, *password creation*, and *drug discovery*?

4. How to incorporate OpenEval in an information extraction system like NELL, and how can it help to improve accuracy of extraction?

2.3.1 Setup

OpenEval is tested on 50 predicates that are chosen randomly from the Freebase knowledge base [Bollacker et al., 2008a], a large semantic database. Freebase contains 360 million instances of different relations. These predicates contain both categories (predicates with one argument) and relations (predicates with two arguments). For each predicate, 25 instances are provided as training seed examples to train OpenEval and 50 instances are randomly chosen as the test data. The test data for each predicate p consists of 25 positive examples (i.e., instances of predicate p) and 25 negative examples. The negative examples are chosen from predicates that are mutually-exclusive to p and also from predicates that are not used as part of training in OpenEval.

Our approach is general and can be used for predicates with any number of arguments. However, to be able to compare our results with other related work, we have tested OpenEval only on predicates with one and two arguments. Tables 2.1 and 2.2 list all the categories, and Table 2 lists all the relations. We have used Support Vector Machines (SVM) as the classification technique in OpenEval. MinorThird [Cohen, 2008], a set of tools for machine learning and information extraction, is used in our implementation.

AcademicField	Airport	Animal	Artery
Attraction	Athlete	Beverage	Bodypart
Color	Conference	Currency	Disease
FilmFestival	Food	Hobby	Language
Muscle	Museum	Protein	Religion
ShoppingMall	Architect	AutomobileModel	
NonProfitOrganization		AwardTrophyTournament	

Table 2.1: List of categories (predicates with one argument) used to train and test OpenEval.

UniversityInCity	CompanyAcquiredCompany	ActorStarredInMovie	AirportInCity
AthletePlaysSport	AutomakerProducesModel	CEOOfCompany	CityCapitalOfCountry
CurrencyOfCountry	DirectorDirectedMovie	DrugHasSideEffect	LanguageOfCountry
MuseumInCity	MusicianPlaysInstrument	NewspaperInCity	PersonGraduatedSchool
TeamWonTrophy	CompanyProducesProduct	RadioStationInCity	SportUsesEquipment
TeamPlaysSport	TransportationInCity	AcademicProgramAtUniversity	
PoliticianHoldsOffice		PlaceOfWorshipPracticesReligion	

Table 2.2: List of relations (predicates with two arguments) used to train and test OpenEval.

2.3.2 Comparison Metrics

We compare our technique to the baselines using the standard performance metrics of precision, recall, and F1. Precision and recall are calculated by:

$$\textit{Precision} = \frac{TP}{TP + FP} \tag{2.8}$$

$$\textit{Recall} = \frac{TP}{TP + FP} \tag{2.9}$$

where True Positives (TP) refer to examples that are correctly labeled as positive, False Positives (FP) are the negative examples that are incorrectly labeled as negative, and False Negative (FN) refers to positive examples that are incorrectly labeled as negative. $F1$ score is defined as:

$$\textit{F1 Score} = \frac{2 * (precision * recall)}{precision + recall}. \tag{2.10}$$

2.3.3 Baseline Approaches

The first baseline that we compare against is the weakly-supervised relation classification [Zhang, 2004]. In this work, a bootstrapping approach is used on top of SVM to classify each predicate instance to one of the predicates in the ontology. The input of SVM is

a set of lexical and syntactic features extracted from the input corpus. To be able to do a fair comparison between Zhang's approach and OpenEval, for each predicate instance in the training and test data, we search the arguments of such predicate instance in Google and crawl the first N returned webpages. N is set to be the same as the number of webpages that are used in OpenEval. The list of predicates, seed examples for each predicate, and webpages crawled from Google are given as an input to Zhang's self bootstrapping approach.

The other baseline for our comparison is the PMI technique. Given a predicate p and a predicate instance i_p, the PMI score is calculated as follows [Turney, 2001]:

$$PMI(i_p, p) = \frac{|Hits(i_p, p)|}{|Hits(p)| \times |Hit(i_p)|} \tag{2.11}$$

where $|$*Hits*$(p)|$ and $|$*Hit*$(i_p)|$ are the number of search engine hits for query p and i_p, respectively, and $|$*Hits*$(i_p, p)|$ is the number of hits when both i_p and p appear in the search query. To evaluate correctness of an instance i_p of predicate p using the PMI technique, we calculate ***PMI***(p, i_p) and if ***PMI***$(p, i_p) > $ ***Threshold***, then we accept i_p as an instance of p.

One should note that PMI is an unsupervised technique and it doesn't require any human annotated data, while OpenEval requires 15-25 seed examples for each predicate. Other studies such as [Etzioni et al., 2004] have shown that the accuracy of the PMI technique can be improved by using patterns (such as "has side effect" for predicate ***DrugHasSideEffect(x,y)***) in the search query. The pattern-based techniques usually require sending a large number of search queries to the search engines. They also require the patterns to be given as an input to the evaluator. In our experiment, the PMI value is calculated only by using the name of the predicate and the predicate instance. Therefore, PMI is taking the advantage of using the predicate name, while OpenEval automatically extracts a set of keywords (i.e., called reference set) that are used to issue search engine queries.

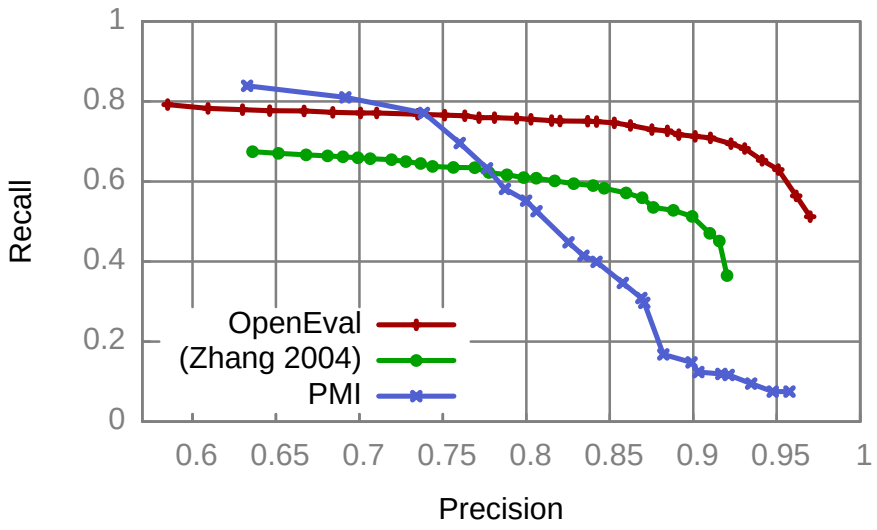


Figure 2.8: The precision/recall curve for test predicate instances. OpenEval uses 100 iterations for training and 5 iterations for evaluation. We also plot the curve for two baseline approaches.

2.3.4 Accuracy of OpenEval in terms of Precision and Recall

Figure 2.8 shows the precision/recall curve of OpenEval, PMI, and Zhang's self bootstrapping approach [Zhang, 2004] for 50 predicates chosen randomly from the Freebase knowledge base. To decide if the input test instance i_p is an instance of predicate p, we check if the correctness probability that is calculated by the OpenEval is greater than a predefined threshold. Different points in the graph are obtained by changing the threshold value for the correctness probability.

The experiments are obtained when OpenEval is trained with 100 iterations and CBIs are extracted by crawling the first 5 webpages from Google. Each iteration of training is a one time step in Algorithm 4 (Lines 9-16) and on average takes 43 seconds in an 8-core 2.67 GHz CPU workstation. The value of parameter C' in predicate evaluator is also set to 3. OpenEval uses 5 iterations (Lines 8-17 in Algorithm 5) to evaluate a new predicate instance.

Figure 2.8 shows that for most of the different recall values, OpenEval achieves signif-

icantly higher precision compared to PMI. Comparing the results of OpenEval and baselines, OpenEval has achieved the best F1 score of 80% while the best F1 scores of PMI and Zhang's approach are around 74% and 70%, respectively. One should note that the F1 score of OpenEval can be improved by using more iterations in the training and evaluation (as will be shown in the following results).

2.3.5 Comparison of Different Values of Parameters

The accuracy of OpenEval depends on the parameters such as the number of iterations that are used in the training. A comparison of different values of these parameters is important for understanding how the result of OpenEval can be improved when more time is given for training.

Figure 2.9 shows the results of OpenEval where the x-axis represents the number of iterations that are used for training. These experiments are done when OpenEval uses only one iteration for evaluation. The result shows that OpenEval achieves a high F1 value even when it uses very few iterations for the training. For example, it achieves a F1 score of about 73% when it uses only 10 iterations. Moreover, the figure shows that the F1 score of OpenEval increases from 0.72 to 0.81, since it uses more numbers of iterations during the training. Note that the number of iterations corresponds to the time that OpenEval uses during the training.

Figure 2.10 shows the F1-score of OpenEval when different numbers of iterations are used for evaluation (OpenEval is trained by using 100 iterations). Interestingly, although the number of training iterations are fixed for all different data points in the figure, the result shows that OpenEval is able to improve its accuracy as more time is given for evaluation. OpenEval uses the trained classifier to measure the effectiveness of each keyword and iteratively finds the set of keywords that are most relevant to the input query. At iteration one, OpenEval achieves a F1 score of 74% while the F1 score is increased to 80% at Iteration 5. It is worth mentioning that each iteration of the predicate evaluator only takes about 1.2 seconds to be completed.

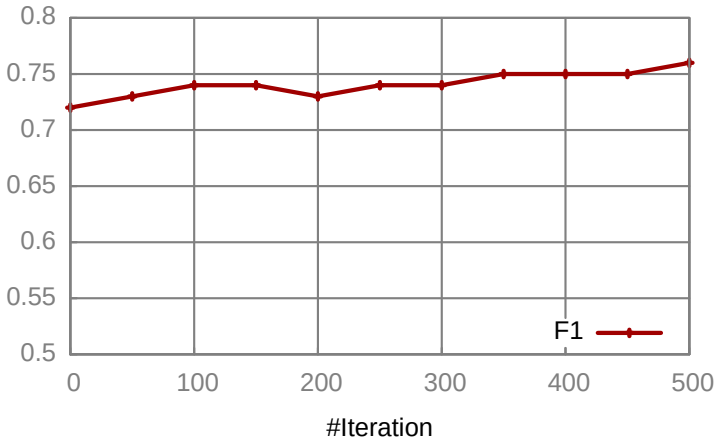


Figure 2.9: F1-score of OpenEval when it uses different numbers of iterations for training.

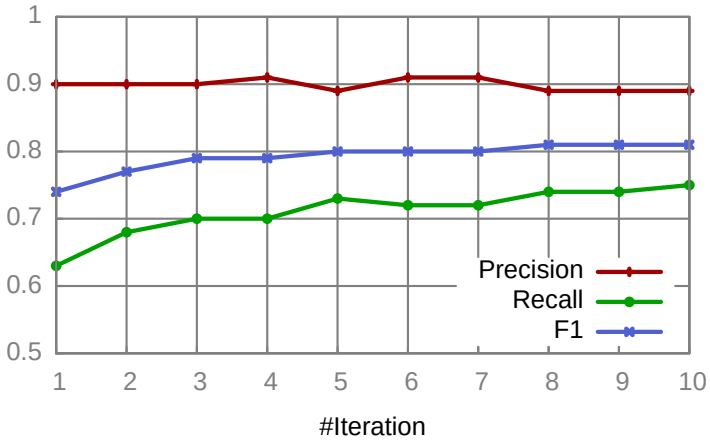


Figure 2.10: Precision, recall, and F1-score of OpenEval when it uses different numbers of iterations for evaluation.

2.3.6 Results of Using OpenEval in Different Applications

We show the detail of the results of using OpenEval in three different applications: *finding objects in an indoor environment*, *detecting picturable and non-picturable words*, and *DrugBank database*. For the first two domains, we show that OpenEval outperforms the baseline approaches and achieves higher accuracy in validating the correctness of a set

of claims. We also show that OpenEval can help to improve the accuracy of DrugBank database.

Finding Objects in Indoor Environment

OpenEval can be used in a mobile robot to determine the name of a place where an object is likely to be located in an indoor environment. This has an important application in robotics since robots must be able to autonomously follow commands that involve finding objects. In Chapter 5, we will explain the details of this approach and will explain how OpenEval is used in this setting to help a real mobile service robot [Samadi et al., 2012; Kollar et al., 2012].

To be able to use OpenEval to find the location of arbitrary objects, we train OpenEval on two predicates that describe where an object can be found in a given location: *objectInLocation(x,y)* and *objectNotInLocation(x,y)*. Each predicate takes the name of an object as the first argument and the location type of the object as the second argument. We collected 914 unique names of objects from Amazon's mechanical Turk for 40 different location types that can be located in different types of indoor environments. Table 2.11 shows all the location types that are used in this experiment.

Apartment	Attic	Balcony	Hospital Ward
Basement	Bathroom	Bedroom	Conference Room
Cafe	Cellar	Classroom	Coffee Shop
Deck	Buffet	Corridor	Computer Lab
Home	Hallway	Gym	Laundry Room
Kitchen	Laboratory	Bar	Living Room
Lobby	Lounge	Nightclub	Waiting Room
Stairway	Prayer Room	Restaurant	Printer Room
Store Room	Study Room	Office	Ticket Office
Kids Room	Home Office		Dining Room

Figure 2.11: List of location types used to train and test OpenEval.

The data is then split by randomly choosing 80% for training and the other 20% for

testing. OpenEval is trained and tested by using the first $N = 20$ webpages that are returned by Bing. OpenEval uses only one iteration for both training and evaluation. We evaluate OpenEval using standard performance metrics of precision, recall, and F1. By using the ESP [von Ahn and Dabbish, 2004] dataset in place of documents retrieved from a search engine, we are able to compare OpenEval to the performance of [Kollar and Roy, 2009].

Figure 2.12 shows the precision/recall graphs for the 40 location types that we used in our experiments. We can see that the model trained on ESP performs worse than OpenEval. This is likely because few of the ESP documents contain location types, so the number of retrieved documents is small.

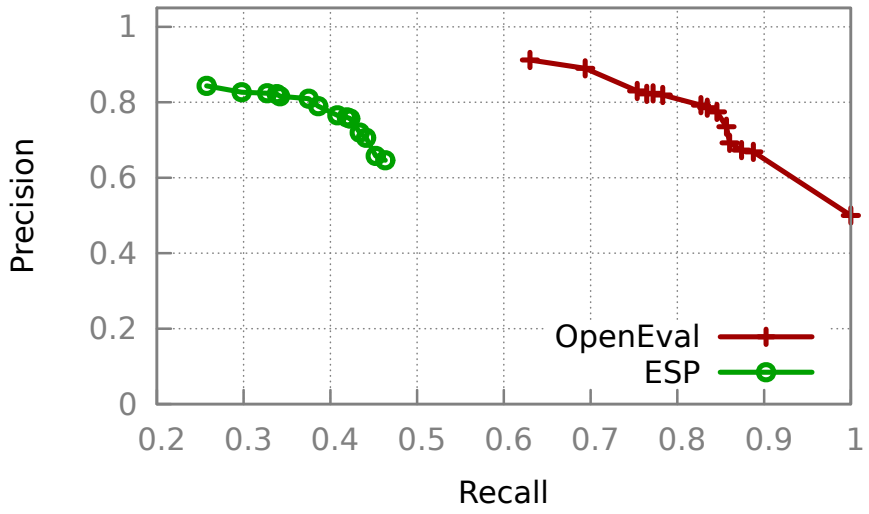


Figure 2.12: The precision/recall curve of the OpenEval approach for over 40 test locations.

Detecting Picturable and Non-Picturable Words

The second domain for which we use OpenEval, is a password management scheme, called *Shared Cues*, developed by Block et al. [Blocki et al., 2013] at Carnegie Mellon University. The main idea behind their technique is to "strategically share cues to make sure that each

cue is rehearsed frequently while preserving strong security goals." In their approach, the user is assigned with an arbitrary object (e.g., a lion), an action (e.g., torturing), and a person (e.g., Michael Jordan). The Person-Object-Action (POA) cue is created by showing the user the picture of the person (e.g., Michael Jordan) and a separate arbitrary picture. The user is then told to imagine that the scene related to the POA is taking place inside the picture, and to create a story that involves the POA. The picture is later shown to the user to help her remember the POA triple. The user password is either the POA triple or is related to it.

In order for the Shared Cues password management schema to work well, it needs to check if a given word (object) is "picturable". A word is labeled as picturable if we are able to draw or find a good image of the word. For example, words such as *car* or *dog* are picturable while *assumption* or *revenge* are not picturable. They have shown that providing a set of picturable words to a user would help her remember the password for a longer time. One of the key challenges for these password-generation schemes is to recognize if a given word is picturable. In this section we show the accuracy of OpenEval when used to evaluate the correctness of predicates *picturableWord(x)* and *picturableWord(x)* where *x* is the input word.

To measure the accuracy of OpenEval in recognizing picturable and non-picturable words, we randomly collected 110 words for each of the predicates *Picturable(x)* and *NonPicturable(x)* from the Basic English Dataset [Ogden, 1944]. We randomly split the data into two sets, 80% for training and the remaining 20% for testing. Figure 2.13 shows the evaluation of OpenEval using metrics of precision and recall. For comparison, we use the ESP dataset[von Ahn and Dabbish, 2004] and classify a word as picturable if it appears as a label of an image in the ESP dataset. OpenEval is trained using 20 webpages returned by the search engine. We have only used one iteration for the training and testing.

DrugBank Database

We apply OpenEval to DrugBank [Knox et al., 2011], a comprehensive online database containing information on drugs, their mechanisms and their targets. DrugBank is a unique bioinformatics and cheminformatics resource that contains detailed drug information. The

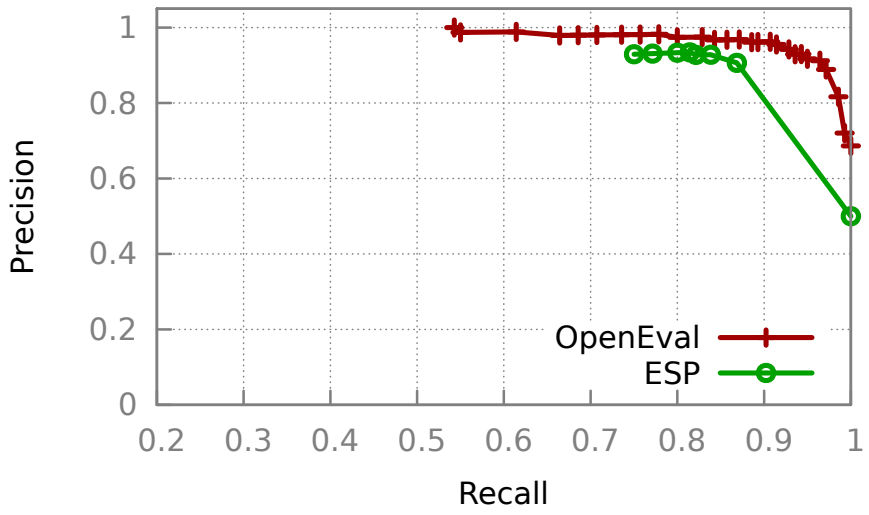


Figure 2.13: The precision/recall curve of the OpenEval approach for predicates *PicturableWord(x)* and *NonPicturableWord(x)*.

thoroughness of DrugBank is demonstrated by the fact that its information is relied upon worldwide by medicinal chemists, pharmacists, physicians, students, and the general public. Over 7,500 drug entries are described in DrugBank (about 1,500 entries are FDA approved; about 6,000 are "experimental"). However, users manually create DrugBank, and therefore drug entries are subject to human error. While DrugBank entries were found to be accurate when an entry is made, by manually checking DrugBank's records, we have found that errors have occurred with updating drug information. For example, some of the drug entries that were FDA-approved upon entry creation were never updated to reflect subsequent FDA withdrawals. A manual audit to detect these errors is not efficient in terms of time and cost, and is subject to human error.

In order to evaluate how OpenEval can improve the accuracy of the DrugBank database, we have chosen 80 drugs. Among these drugs, 40 of them are approved, and others are withdrawn from the market. All of those 80 drugs are used as the training data. For the test data, we have (manually) found 12 drugs that are listed as approved drugs in DrugBank 2013 [Knox et al., 2011] but are withdrawn from either the Europe, Canada, Australia, or US market. We have also used another set of 12 approved drugs as part of the test data.

OpenEval is trained and tested using 50 webpages returned by the Bing search engine. We have only used one iteration for the training and testing. During the training, we have explicitly excluded the DrugBank website from the sources that OpenEval may use for retrieving information from the Web.

Figure 2.14 shows the precision recall of OpenEval on the test data chosen from the DrugBank database. The figure shows that OpenEval achieves precision of 0.84 at the recall value of 0.4. The results indicate that OpenEval can improve the manual process of checking the DrugBank data, by automatically checking the accuracy of some of the entries in the DrugBank database and finding some of the data that are likely to be false. For example, if we set the threshold of OpenEval to 0.6, then four out of five drugs that have a confidence value of less than 0.6 are officially withdrawn from the market: *Temazepam* (withdrawn from Australian market), *Levamisole* (withdrawn from the U.S. and Canadian markets), *Rimonabant* (withdrawn from the European market), and *Lumiracoxib* (withdrawn from the Australia, UK, and other markets).

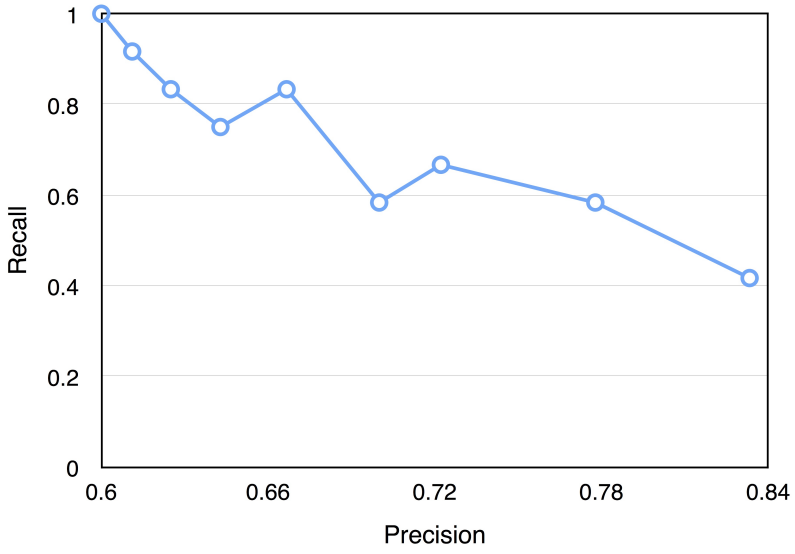


Figure 2.14: The precision/recall curve of the OpenEval approach for approved and withdrawn drugs. The test data consists of 12 approved and 12 withdrawn drugs. The withdrawn drugs are listed as "approved" in the DrugBank database 2013 [Knox et al., 2011], but are withdrawn by official sources (these are errors in the database).

2.3.7 Using OpenEval as part of the NELL System

Never Ending Language Learning (NELL) [Carlson et al., 2010a; Mitchell et al., 2015a] [2], is a never-ending learning agent whose goal is to learn to read the Web. The input of NELL is: (i) an ontology that consists of a set of one- or two-argument predicates, (ii) 10-20 labeled training examples for each predicate, (iii) the Web data, and (iv) occasional human labeled data (i.e., a set of labeled predicate instances). Given this input, NELL is run 24 hours 7 days per week, and either extracts a set of new predicate instances or corrects its beliefs on the predicate instances that are already extracted. In addition, NELL learns how to read better over time. NELL has been running since January 2010, and has extracted over 80 million predicate instances.

NELL's learning tasks consists of: *category classification*, *relation classification*, *entity resolution*, and *learning inference rules*. In this chapter, we only focus on the first two learning tasks, since OpenEval is directly used in both of these tasks to improve the accuracy of NELL.

NELL's goal for *category extraction* task is to extract a noun phrase and map it to one or some of the categories in its ontology. For each category, NELL uses five distinct functions, where each function predicts if a noun phrase belongs to a category. These functions include: Coupled Morphological Learner (CML) [Carlson et al., 2010d], Coupled Pattern Learner (CPL) [Carlson et al., 2010d], Set Expander for Any Language (SEAL) [Wang and Cohen, 2009], Never Ending Image Learning (NEIL) [Chen et al., 2013b], and OpenEval. For the task of *relation extraction*, NELL uses CPL, SEAL, and OpenEval to classify if a given pair of noun phrases belong to a relation. NELL's software architecture is shown in Figure 2.15 (figure is reprinted from [Mitchell et al., 2015a]).

We evaluate the effectiveness of OpenEval in the NELL system by showing the official result of the Knowledge Base Population (KBP) evaluation[3], which was organized by the Text Analysis Conference in 2013. The goal of KBP is to evaluate technologies for building and populating knowledge bases that are built by automatically extracting named entities from unstructured text. NELL was participating in the Slot Filling (SF) task of

[2] http://rtw.ml.cmu.edu/

[3] http://www.nist.gov/tac/2013/KBP/

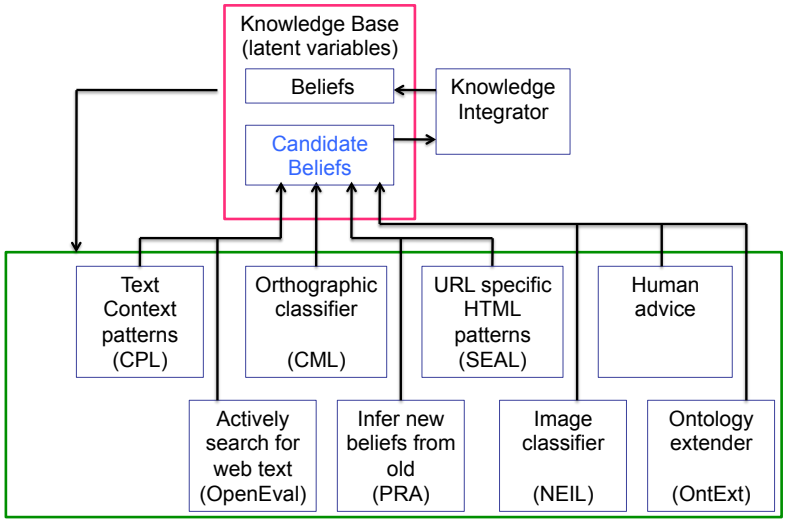


Figure 2.15: NELL's software architecture. Reprinted from [Mitchell et al., 2015a].

KBP in 2013. We first explain the detail of the SF task and then explain how OpenEval helps NELL to improve its accuracy for this particular task. Second, we compare the result of OpenEval with other components of the NELL system using a set of random predicates and predicate instances obtained from the Freebase knowledge base.

For the KBP Slot Filling task, an initial knowledge which is a snapshot of English Wikipedia is provided as an input. Each page in the KB corresponds to a Wikipedia page for either a person, an organization, or geopolitical entity. Each of these entities have a set of predefined attributes (a.k.a *slots*) which are derived from Wikipedia infoboxes. The goal of the KBP Slot Filling task is to collect information from the input corpus regarding certain attributes about target entities in unstructured text. For example, for a person, a slot can be a single-argument predicate *DateOfBirth* or two-argument predicate *EmployeeOf*. The list of attributes for entity types *person* and *organization* are listed in Tables 2.16 and 2.17, respectively.

CMUML [Bryan Kisiel, 2013] is the name of the CMU system for the KBP 2013 English Slot Filling (SF) task. CMUML was using a combination of distant supervision [Mintz et al., 2009], generalized stacking [Wolpert, 1992a] and CRF-based structured prediction [Betteridge et al., 2014; Krishnamurthy and Mitchell, 2012]. Given a candidate

AlternateNames	DateOfBirth	Age	CountryOfBirth
StateOrProvinceOfBirth	CityOfBirth	Origin	DateOfDeath
CountryOfDeath	Title	CityOfDeath	CauseOfDeath
CountriesOfResidence	Charges	CitiesOfResidence	SchoolsAttended
StateOrProvinceOfDeath	Children	Religion	Spouse
EmployeeOrMemberOf	Parents	Siblings	OtherFamily
StatesOrProvincesOfResidence			

Figure 2.16: Predicates (slot names) used for entity person in KBP 2013 evaluation.

CountryOfHeadquarters	FoundedBy	TopMembersEmployees	Members
NumberOfEmployeesMembers	MemberOf	AlternateNames	Parents
PoliticalReligiousAffiliation	DateFounded	DateDissolved	Subsidiaries
StateOrProvinceOfHeadquarters	Shareholders	CityOfHeadquarters	Website

Figure 2.17: Predicates (slot names) used for entity organization in KBP 2013 evaluation.

sentence, the CMUML system first generates different layers of annotation using multiple syntactic and semantic annotators. A Conditional Random Field [Lafferty et al., 2001] (CRF)-based structure predictor is then used to integrate and map these layers to the KBP ontology. Finally, the extractions from the CRF were integrated and validated before producing the final answers. The detail of the CMUML system is explained in [Bryan Kisiel, 2013] and the overall system architecture is shown in Figure 2.18.

Slot filler values that are extracted by the CMUML system are evaluated and filtered by OpenEval to determine whether or not sufficient evidence for them could be found by querying the live Web. Note that the official submission of the CMUML system did not use the OpenEval component, since web access was not allowed in the main submission. However, we were allowed to submit unofficial versions of the system for evaluation.

Table 2.3 shows the official result published by KBP organizers. The table shows the result of two different CMUML submissions, where *CMUML+OpenEval* builds on *CMUML* by including OpenEval as the post-processing validation step. The table shows

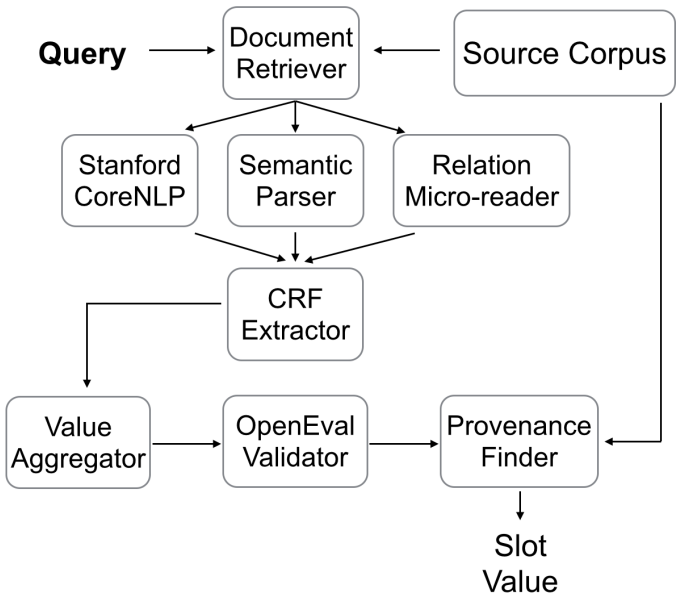


Figure 2.18: Overview of the CMUML Slot Filling (SF) system. Reprinted from [Bryan Kisiel, 2013].

System	Recall	Precision	F1
CMUML	9.67	30.34	14.67
CMUML+OpenEval	10.69	32.3	16.07

Table 2.3: Official evaluation scores of various CMUML submissions. *CMUML* is the system based on NELL, and *CMUML+OpenEval* uses OpenEval as the post-processing validation step.

that OpenEval greatly improves both the precision and recall of the CMUML system. OpenEval improves the precision of CMUML by 2% without any cost in the recall. Indeed, the recall of *CMUML+OpenEval* is improved by 1%, compared to the CMUML system. Overall, the F1 score of *CMUML+OpenEval* is improved by 1.6%.

2.4 Summary

This chapter introduced OpenEval, a novel online information validation approach that automatically evaluates the correctness of a predicate instance using the Web. We experimentally showed that OpenEval massively outperforms the related techniques such as PMI and weakly-supervised relation classification technique [Zhang, 2004]. We also discussed the detail of the results of using OpenEval in four different applications. Overall, OpenEval is a general anytime technique, requires minimum supervision (in terms of seed examples and input ontology), and improves its accuracy as more time is given for evaluation and training.

Chapter 3

Knowledge Integration and On-Demand Time-Budgeted Query Answering

Up to now, we have assumed that only a single knowledge harvesting system, OpenEval, is available to process the information. In this chapter, the assumption is changed so that there are multiple knowledge harvesting systems available to be queried. We address the problem of integrating knowledge from all of the systems. We also enable an answer of a query to be calculated by requesting and aggregating the opinion of different systems for different *non-query* predicates. Finally, we address the problem of how to best utilize the input time-budget by choosing which queries to send to which resources.

We begin in Section 3.1 with motivating the problem of integrating opinions from multiple knowledge harvesting systems, and requirements that a knowledge-on-demand system should accommodate. Section 3.2 formally defines the problem of on-demand time-budgeted query answering. Section 3.3 presents the details of the AskWorld approach and the formulation of the problem with MDP. Section 3.4 reports our experimental results and compares AskWorld to baseline and state-of-the-art approaches. Section 3.5 summarizes this chapter.

3.1 Introduction

One of the main motivations behind the OpenEval system, described in Chapter 2, was to develop an information extraction approach that achieves high recall in providing accurate answers to queried predicate instances, by processing unstructured information on the Web. In addition to OpenEval, over the past few years, several other knowledge bases (KBs) and information extractors have been developed, examples include NELL [Carlson et al., 2010a], Yago [Suchanek et al., 2007], and Freebase [Bollacker et al., 2008b]. These KBs can cover thousands of predicates (e.g., *HealthyFood*, *FoodCanCauseDisease(Food, Disease)*) and millions of instances of such predicates (e.g., *Sugar* is an *UnHealthyFood*). While some of these KBs are user contributed (e.g., Freebase), others are constructed from semi-structured data (e.g., Yago), or from unstructured Web data (e.g., NELL).

These knowledge harvesting systems vary significantly in the *type of the data* and *techniques* that they use, and therefore in their precision, recall, and delay in responding to different types of queries. For example, Freebase is primarily built by its community members and therefore is highly precise, although its coverage can be limited. For example, 70% of people included in Freebase have no place of birth information [West et al., 2014b]. In contrast, NELL has a different ontology covering somewhat different categories and relations, including many informal relations such as BeverageServedWithBakedGood(x,y). OpenEval automatically extracts information on demand from both structured and unstructured information on the Web and therefore achieves higher coverage at the price of losing some precision. For all of these systems, precision and recall also can vary dramatically for different predicates, e.g. NELL is more accurate for categories compared to relations. The different systems also have very different response times. For example, querying a static knowledge base such as Yago involves only a quick database lookup, while querying OpenEval takes longer since it crawls and parses information from the Web as needed to attempt to answer the query on demand. Together, we shall refer to these KBs and on-demand extractors as *Knowledge Resources (KRs)* in the rest of this chapter.

Given the heterogeneity of source data and extraction algorithms involved, these KRs can contain complimentary or often conflicting facts. Moreover, the degree of these differences and expertise of each KR may vary from one predicate to another. In order to

satisfy the knowledge need of an end user or application [Samadi et al., 2012], a single KR is not sufficient and it is necessary to integrate evidence from all these different KRs and return a consolidated response. For example, a real mobile service robot may need to query the Web and fill up its missing knowledge required to perform a task. Depending on the task that the robot/agent is performing, it may need different types of knowledge (instances of different predicates), and responses within different time budgets. In other words, it needs to have access to a *Knowledge-On-Demand (KoD)* service which is able to aggregate opinions from all these diverse KRs taking their respective complementarity, conflicts, and expertise into account.

3.1.1 Requirements that a Knowledge-On-Demand Systems Should Accommodate

There are two requirements that such KoD systems should accommodate.

A. Knowledge integration: Even for a given entity, the KI may request multiple opinions from the same or different expert component, and hence should be able to automatically integrate all the responses. Predicates in the KRs often have coupling relationships among them. For example, even though the input query asks whether *sugar* is an *UnHealthyFood*, the KI may probe OpenEval to check whether it thinks *sugar* might also be a *HighSucroseFood*, another predicate from AskWorld's ontology. Since *HighSucroseFood* and *UnHealthyFood* are usually correlated, OpenEval's opinion on such a *non-query predicate* (*HighSucroseFood*) may ultimately influence AskWorld's response on the *query predicate* (*UnHealthyFood*). As we shall see in Section 3.4.2, significantly more accurate responses are obtained when KR opinions on non-query predicates are also aggregated.

B. Time-budgeted query answering: While the result of expanding query evaluation is promising, it puts additional constraints on how best to utilize and allocate the already scarce time budget specified by the user or application making the query. As the second requirement, the user (or application) generally needs to obtain an answer within a bounded time budget that may vary with each query. AskWorld, therefore, assumes that its input includes both a *query* and a *time budget*, and attempts to provide a best-effort response making full use of the available budget, with a potentially improved response with

increasing budgets.

Ideally, given an input query such as *"Is City(Buenos Aires) true?"*, we would like the KoD service to aggregate opinions from all available KRs. Response time from a KR may vary depending on the predicate, especially for on-demand extractors such as OpenEval. In many applications of practical significance, the final response is desired within a specified time budget, and polling all available KRs is unfortunately infeasible. Thus, the KoD service has to devise a *policy* to decide on which subset of KRs to poll as a function of the query predicate and the specified time budget. We emphasize that the time budget may vary depending on the tolerance limits of the agent using the KoD service, and we would like the policy to be able to incorporate this gracefully, returning a more accurate response when more time budget is available. The polling policy also needs to be prepared a-priori (as there may not be time to learn the policy on the fly).

Recently, the problem of learning under budget-constraints has received considerable attention [Saberian and Vasconcelos, 2010; Xu et al., 2012, 2013; Karayev et al., 2013]. We review these techniques in Chapter 7. While some aspects of the problems mentioned above have been studied in previous research, to the best of our knowledge, no previous research has addressed all the issues simultaneously, and definitely not in the context of a KoD service. We bridge that gap in this chapter and propose the AskWorld system.

3.1.2 AskWorld

AskWorld, the system proposed in this chapter, aims to specifically satisfy these information needs by learning a policy for making subqueries to a diverse collection of available resources, attempting to optimize the query response it can provide within the allotted time budget, and integrating the results returned by the different resources. The architecture of the AskWorld system is shown in Figure 3.1. Given a user query with associated time budget, the Budget Policy Executor (BPE) module of AskWorld selects the subset of available KRs to poll, such that the ultimate response can be provided within the specified time budget. Responses from the KRs are integrated by the Knowledge Integrator module and returned to the user.

AskWorld incorporates an on-demand Knowledge Integrator (KI) that aggregates opin-

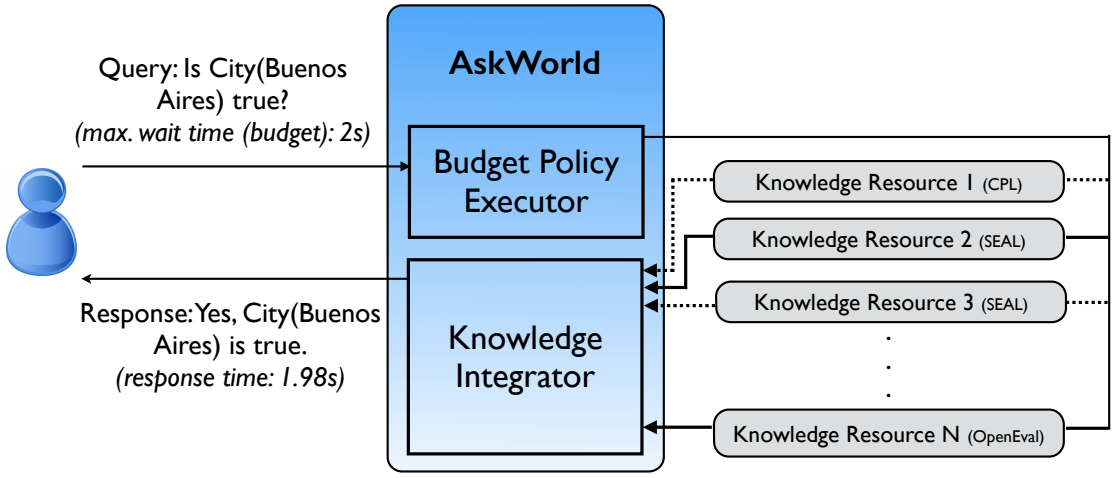


Figure 3.1: Architecture of the AskWorld system. Given a user query (e.g., *UnHealthyFood(sugar)?*) and a time-budget within which this query must be answered (e.g., 2sec above), the Budget Policy Executor selects a subset of the available knowledge resources (KRs) to poll (shown by solid lines in the figure; dotted lines connect knowledge resources that are not used to answer this particular query at the specified time budget). Responses from the KRs are integrated by the Knowledge Integrator and the final response is returned to the user within the specified time budget.

ions from other existing KBs (e.g., YAGO), and IE systems such as s NELL. The AskWorld architecture is shown in Figure 3.1, where CPL, CMC, and SEAL are three NELL subsystems that AskWorld's KI has access to, and OpenEval is an external information extraction technique that AskWorld can query. We can think of each one of them as an expert whose opinions are integrated by the KI. For example, given the entity *Buenos Aires* and predicate *City*, CPL will explore in unstructured text whether it can find patterns of the form *"is the capital of"*, and if it does, then its confidence in *Buenos Aires* being a *City* increases, which is then communicated to the KI. Similarly, the CMC sub-component looks for certain orthographic features in the entity name which might indicate that the entity as an instance of the corresponding predicate. Similarly, the SEAL sub-component checks whether *Buenos Aires* is redundantly mentioned in the context of other known unhealthy foods in HTML tables on the Web. OpenEval automatically generates the Google search

query "Buenos Aires city" and evaluates if *Buenos Aires* is an instance of the*City* predicate by parsing and processing the search results. In short, given an entity and a predicate, each of these components will return their confidence in the entity being an instance of the predicate. These returned confidences are then integrated by the KI and the final response is returned to the user or application. Thus, by exploring such components and existing KBs, AskWorld is able to provide a *knowledge-on-demand (KoD)* service.

While some aspects of the problems mentioned above have been studied in previous research and shall be reviewed in Chapter 7, to the best of our knowledge, no previous research has tackled all the issues simultaneously, and definitely not in the context of a KoD service.

3.2 Problem Statement

Let $(p, x, \mathcal{B})$ be a user query, where p is a predicate from an ontology $\mathbf{O}$, x is a candidate instance, and $\mathcal{B}$ is a response time budget (e.g., *(isCity, Buenos Aires, 2sec)*). We want AskWorld to validate whether x is a true instance of p by classifying it within the time budget $\mathcal{B}$ to one of the labels from $\mathcal{Y} = \{false, true\}$. AskWorld may poll a set of knowledge resources (KRs), $\mathbf{R}$, to determine whether x is an instance of p or of any other predicates from the ontology $\mathbf{O}$, and aggregate all of the responses. The poll (p', x, r) checks the opinion of resource $r \in \mathbf{R}$, on whether x is an instance of predicate $p' \in \mathbf{O}$ and costs $c(r, p')$ time (note that p' is not necessarily equal to p). We assume that $c(r, p')$ remains constant during train and query time, and for the entire training data, we extract the value of all the queries during the training time (regardless of the query cost). We also assume that an upper bound on the test budget is known during training, but the exact value of the test budget may vary for different test instances.

To simplify our explanation, we assume that we have a set of polling queries $\mathbf{K} = \{\langle r, p' \rangle \mid r \in \mathbf{R}, p' \in \mathbf{O}\}$, i.e., all possible combinations of knowledge resources in $\mathbf{R}$ and predicates in $\mathbf{O}$. For each possible combination of resources in $\mathbf{R}$ and predicates in $\mathbf{O}$, there is a correspondence query $q = \langle r, p \rangle$ in $\mathbf{K}$. In this case, $|\mathbf{K}| = |\mathbf{R}| \times |\mathbf{O}|$. Cost of query q, denoted by $c(q)$, is equal to $c(r, p)$.

Given a user query, the main challenge is to learn a policy that identifies a subset of resource-predicate polling queries $\mathcal{K} \subseteq \mathbf{K}$ so that the most accurate response is provided within the response time budget. The policy should be able to handle varying query-time budget (subject to a maximum upper bound) without the need for retraining.

Note that while the assumption that the cost of polling a KB with a predicate instance is constant during train and test time may not be always true, we think that this is a reasonable assumption in the setting explored in this chapter. To verify whether this assumption is true or false, we have queried each knowledge resource 10K times at different times of the day. Average query cost (millisecond) and standard deviation of query cost for each knowledge resource are as follows: KB: (AVG=77.5, STD=3.7) , CKB: (AVG=84.2, STD=6.27), CMC: (AVG=90.8, STD=5.0). This validates our claim that the cost of polling a KR for test and training times are similar, since the standard deviation is relatively low compared to the average values. We leave it to future work to extend AskWorld to handle cases where the test and training times are different.

3.3 Our Approach

In this section, we describe the AskWorld system. The AskWorld (Algorithm 6) consists of two steps. In step 1, given a query and a time budget, AskWorld identifies the subset of knowledge resources to poll using BUDGET POLICY EXECUTOR (BPE) (Section 3.3.1); and in step 2 it aggregates the responses obtained from step 1 using KNOWLEDGE INTEGRATOR (Section 3.3.3) and returns the final result to the user.

3.3.1 Markov Decision Process (MDP) Formulation of BUDGET POLICY EXECUTOR (BPE)

We cast the budget-sensitive query evaluation problem as solving a Markov Decision Process (MDP), $\mathcal{M}$, represented as a tuple $\mathcal{M} = \langle \gamma, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ where γ is a discounting factor. The remaining components of the MDP are described below:

Algorithm 6 AskWorld - Query evaluation for knowledge-on-demand

Require: $\langle p, x, \mathcal{B}, Q, h \rangle$ /* p: predicate, x: instance to be evaluated, $\mathcal{B}$: input budget, Q: Budget policy learned during training h: knowledge integrator function. */

/* **Step 1: BudgetPolicyExecutor** */

$t \leftarrow 0$; $\mathcal{B}_0 \leftarrow \mathcal{B}$; $\mathcal{K}_0 \leftarrow \{\}$

$S_0 \leftarrow \langle \mathcal{K}_0, \mathcal{B}_0 \rangle$

while $\mathcal{B}_t > 0$

// use budget policy Q to select next resource-predicate pair, $\langle r, p' \rangle$, to poll

$\boldsymbol{a}_{\langle \boldsymbol{r}, \boldsymbol{p'} \rangle} \leftarrow argmax_{a_{\langle r, p' \rangle}} Q(S_t, a_{\langle r, p' \rangle})$

Poll resource $\boldsymbol{r} \in \mathbf{R}$ to validate if x is an instance of predicate $\boldsymbol{p'}$

// note that $\boldsymbol{p'}$ is not necessarily equal to p

$\mathcal{K}_{t+1} \leftarrow \mathcal{K}_t \cup \{\langle \boldsymbol{r}, \boldsymbol{p'}(x) \rangle\}$ // update poll response set

$\mathcal{B}_{t+1} \leftarrow \mathcal{B}_t - c(\boldsymbol{r}, \boldsymbol{p'})$ // update residual budget

$S_{t+1} \leftarrow \langle \mathcal{K}_{t+1}, \mathcal{B}_{t+1} \rangle$ // update state

$t \leftarrow t + 1$

end while

/* **Step 2: KnowledgeIntegrator** */

Build feature vector x' and indicator vector z from the result of polls in $\mathcal{K}_t$

return $\mathbf{y}^* = \arg\max_{y \in \{\textit{true}, \textit{false}\}} h(x', y, z)$

States ($\mathcal{S}$): Each state, $\mathcal{S}_t = \langle \mathcal{K}_t, \mathcal{B}_t \rangle \in \mathcal{S}$, of the MDP represents intermediate knowledge acquired, where $\mathcal{K}_t \subset \mathbf{K}$ is the set of knowledge resource-predicate polling queries issued by AskWorld and responses received, and $\mathcal{B}_t$ is the residual budget allowed to be used starting from state $\mathcal{S}_t$. During query time, the agent starts executing the MDP policy starting from $\mathcal{S}_0 = \langle \{\}, \mathcal{B} \rangle$, where $\{\}$ indicates that no polling response (feature) has been acquired and $\mathcal{B}$ is the input budget.

Actions ($\mathcal{A}$): Each action $a_{\langle r, p' \rangle} \in \mathcal{A}$ corresponds to a resource-predicate tuple $\langle r, p' \rangle$ which indicates which knowledge resource r to poll to validate if x is an instance of predicate p'. By taking the action $a_{\langle r, p' \rangle}$, we acquire the response from resource r at the cost of $c(r, p')$.

Transition function ($\mathcal{T}$): The transition function, $\mathcal{T}(\mathcal{S}_t, a_{\langle r,p'\rangle}, \mathcal{S}_{t+1})$, is defined as follows:

$$\mathcal{T}(\mathcal{S}_t, a_{\langle r,p'\rangle}, \mathcal{S}_{t+1}) = \begin{cases} 1 & \text{if } \mathcal{B}_{t+1} = \mathcal{B}_t - c(r,p') \geq 0 \\ & \text{\& } \langle r,p'\rangle \in \mathcal{K}_{t+1}, \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\mathcal{B}_t$ is the remaining budget that we are allowed to use starting from state S_t. Intuitively, Equation 3.1 says that the probability of moving from state $\mathcal{S}_t$ to $\mathcal{S}_{t+1}$ by taking action $a_{\langle r,p'\rangle}$ is equal to 1 if the cost of $a_{\langle r,p'\rangle}$ is less than the remaining budget $\mathcal{B}_t$ and if the query $\langle r,p'\rangle$ is part of $\mathcal{K}_{t+1}$.

Why should budget be included in the transition function?
The alternative option, similar to [Weiss and Taskar, 2013a], is to incorporate the budget constraint as part of the reward function. For example, when a transition is invalid (i.e., we are out of budget), the value of the reward function would be either a large negative value or zero. This approach has two main disadvantages: (i) defining the reward to be negative for the invalid transitions may affect the learning process since the expected reward value of a state can be potentially penalized as the agent receives negative value for the states that it is not able to move to, and (ii) defining the reward of the invalid transitions to be equal to zero also makes it impossible for the agent to differentiate between states that are not reachable and those that have the expected reward of zero. By defining the budget constraint as a part of the transition function, we avoid both of these problems since the agent is not able to traverse to states that require more budget than the remaining budget $\mathcal{B}_t$. Thus, unlike [Weiss and Taskar, 2013a]'s approach which may find a policy that violates the input budget, our approach always finds a policy that satisfies the input budget constraint.

Reward function ($\mathcal{R}$): The reward function $\mathcal{R}$ is defined as the value of the information added by each action $a \in \mathcal{A}$ given current state $\mathcal{S}_t$. More precisely,

$$R(\mathcal{S}_t, a, \mathcal{S}_{t+1}) = \begin{cases} \frac{1}{|\Psi'|} \sum_{(x_i,y_i)\in\Psi'} \bar{h}(x_i,y_i,z_{t+1}) - \bar{h}(x_i,y_i,z_t) & t \neq 0 \\ 0 & t = 0 \end{cases} \tag{3.2}$$

where Ψ' is the set of evaluation data (subset of the input training data Ψ) in the form of $\Psi_i = (x_i, y_i)$. z_t is defined as a binary vector where $|z_t| = |\mathbf{K}|$. The jth element of z_t is

equal to 1 if the value of query $\langle r, p\rangle$ appears in the state $\mathcal{S}_t$ (as part of $\mathcal{K}_t$), and otherwise it is equal to 0. Vector z can be seen as an indicator vector used as input of a predictor (knowledge integrator, Section 3.3.3) function. Function $\bar{h}(x, y, z)$ is defined in terms of $h(x, y, z)$. Function $h(x, y, z)$ is a predictor function which returns the confidence value of predicting label $y \in \mathcal{Y}$ for the input instance x using queries indicated by vector z. The function $\bar{h}(x, y, z)$ is defined as follows:

$$\bar{h}(x, y, z) = h(x, y, z) - \max_{y' \neq y} h(x, y', z) \tag{3.3}$$

Function $\bar{h}$ returns the maximum difference in the confidence value of the classifier in predicting the true label compared to all the other labels in $\mathcal{Y}$.

Intuitively, Equation 3.2 says that each time that we poll $\langle r, p\rangle$, the reward that we receive is equal to the change in the margin of our predictor h, averaged over all the training data points. In other words, the reward function measures the value of the knowledge that we acquire from each of the resources. If the value of $R(\mathcal{S}_t, a, \mathcal{S}_{t+1}) > 0$, then it means that the selected action increases the confidence value of our predictor in predicting the true label compared to the other labels, and if $R(\mathcal{S}_t, a, \mathcal{S}_{t+1}) < 0$, it means that it is decreasing the confidence value. Ideally, we would like to choose a polling query that increases the confidence of our prediction when moving from state $\mathcal{S}_t$ to $\mathcal{S}_{t+1}$.

Given a deterministic policy Π and a sequence of states $\mathcal{S}_0, ..., \mathcal{S}_n$ computed from Π, we define $\mathcal{R}_\Pi$ as the reward that we receive by the following policy determined by Π. Then,

$$\mathcal{R}_\Pi = \sum_{(x_i, y_i) \in \Psi'} \bar{h}(x_i, y_i, z_n) - \bar{h}(x_i, y_i, z_0) \tag{3.4}$$

Equation 3.4 says that the reward that we receive from following policy Π is equal to the difference between the confidence value of the predictor h when using the acquired queries in state $\mathcal{S}_n$ and the queries in state $\mathcal{S}_0$.

Predictor Function: To build the predictor function $h(x, y, z)$, we use Support Vector Machines (SVM). The SVM is trained by assuming that all the polling query responses (feature values) are acquired for all the training instances. For any new instance where some of the feature values are missing (i.e., zero elements in the z vector), we use the prior value for the missing feature as the average feature value over all the training data (assuming that the size of positive and negative data are equal). Thus, each feature of the

SVM corresponds to a response to a polling query in $\mathbf{K}$, thereby making its feature space $|\mathbf{K}|$-dimensional.

Theorem 1. *(***Property of MDP***) In the MDP $\mathcal{M}$, for every two policies Π and Π' which respectively map states $\langle\{\}, \mathcal{B}\rangle$ and $\langle\{\}, \mathcal{B}'\rangle$ ($\mathcal{B} \neq \mathcal{B}'$) to some actions, $\mathbf{S}(\Pi) \cap \mathbf{S}(\Pi') = \varnothing$, where $\mathbf{S}(\Pi)$ and $\mathbf{S}(\Pi')$ are the sets of all the states that can be generated by policies Π and Π', respectively.*

This theorem can be proved by contradiction. Assume that in separate executions of policies Π and Π', starting from initial states $\mathcal{S} = \langle\{\}, \mathcal{B}\rangle$ and $\mathcal{S}' = \langle\{\}, \mathcal{B}'\rangle$ ($\mathcal{B} \neq \mathcal{B}'$), we could reach to states $\mathcal{S}_1 = \langle\mathcal{K}_1, \mathcal{B}_1\rangle$ and $\mathcal{S}_2 = \langle\mathcal{K}_2, \mathcal{B}_2\rangle$, where $\mathcal{S}_1 = \mathcal{S}_2$. In other words, we assume that there is at least one common state that separate executions of policies Π and Π' may reach to, i.e. $\mathcal{S}_1 \in \mathbf{S}(\Pi) \cap \mathbf{S}(\Pi')$ and $\mathcal{S}_2 \in \mathbf{S}(\Pi) \cap \mathbf{S}(\Pi')$. Both states $\mathcal{S}_1$ and $\mathcal{S}_2$ are identical, and hence $\mathcal{K}_1 = \mathcal{K}_2$ and $\mathcal{B}_1 = \mathcal{B}_2$. From Equation 3.1, we know that $\mathcal{K}_1$ and $\mathcal{K}_2$ contains all the actions (i.e., polling a knowledge resource for a predicate) that are executed from the initial states $\mathcal{S}$ and $\mathcal{S}'$. Since, $\mathcal{K}_1 = \mathcal{K}_2$ and $\mathcal{B}_1 = \mathcal{B}_2$, we can reverse all the actions in $\mathcal{K}_1$ and $\mathcal{K}_2$, add the corresponding budget to $\mathcal{B}_1$ and $\mathcal{B}_2$, and reach to the same initial states with the same initial budget cost, which is in contradiction to our assumption that $\mathcal{B} \neq \mathcal{B}'$.

Intuitively, Theorem 1 says that the policies we learn for the different initial budgets are independent of each other. In other words, there is no advantage of simultaneously learning policies for the different initial budget values. In the next section, we explain how to address this problem by abstracting states in the MDP.

3.3.2 Solving MDP

The number of states in the MDP $\mathcal{M}$ depends on the size of ontology, number of resources, and the upper bound on the input test budget. Depending on the number of predicates in the ontology $\mathbf{O}$ and the resources in $\mathbf{R}$, the state space can be very large. For example, for an ontology of size 25 and 3 resources, the size of MDP is of the order of 2^{75}. Given the size of the state space, it is practically infeasible to directly learn a policy for each state of the MDP.

In this section, we present two approaches to solve the MDP and will compare their results in the experimental results section. First, we explain how to directly abstract the state space in the MDP and solve the abstracted MDP using the value iteration algorithm. We then explain how to learn the policy using Q-Learning with linear function approximation [Sutton and Barto, 1998]. We also discuss the advantages and the disadvantages of each of these approaches.

AskWorld (V*): Abstracting MDP and Solving It Using Value Iteration

We abstract the state space of $\mathcal{M}$ to a smaller space. The resulting MDP is denoted by $\bar{\mathcal{M}}$. The abstraction is done in the following way: (i) defining the remaining budget in each state to be within an interval, instead of being equal to an exact budget amount, and, (ii) reducing the size of polling queries in $\mathbf{K}$ to a smaller set $\bar{\mathbf{K}}$.

To define the budget interval for each state, we assume that $\mathbf{B}_U$ is the upper bound on the budget that we receive during query time. The size of each interval is defined to be equal to δ and we have $\frac{\mathbf{B}_U}{\delta}$ distinct intervals. In the abstract MDP, each state is defined as $\mathcal{S}_t = \langle \mathcal{K}_t, [\mathcal{B}_t^L, \mathcal{B}_t^U] \rangle$, where the second element is the budget interval, $\mathcal{B}_t^L = k \times \delta$, and $\mathcal{B}_t^U = (k+1) \times \delta$. The reward function in the abstract MDP is the same as the original MDP.

The transition function for the abstract MDP depends on the budget interval defined in each state, the cost of the actions, and the value of the budget that we receive during the query time. For example, assume that during the test time, we are in a state where the budget interval is $[40, 60]$, $\delta = 20$, and we are executing action $a_{\langle r,p \rangle}$. If the true remaining budget is 40 and $c(a_{\langle r,p \rangle}) = 5$, then we move to a state where the budget interval is $[20, 40]$. However, if the remaining budget is 50, we then move to a state where the budget interval is $[40, 60]$ (in both these cases $\langle r, p \rangle$ is added as one of the queries in the state). The problem is that we don't know the exact budget that is given during the test time while learning a policy for the MDP. To address this issue, we assume that different budget values are equally likely to be given during test time, and then define different probabilities for transitioning to different states, e.g. in our example, we move to the state with budget interval $[20, 40]$ with probability $\frac{c(a_{\langle r,p \rangle})}{\delta} = 0.25$ and the move to the state with budget

interval $[40, 60]$ with probability $\frac{\delta - c(a_{\langle r,p \rangle})}{\delta} = 0.75$. This example helps us to formally define the transition function.

To define the transition function, we know that the cost of each action $a_{\langle r,p \rangle}$ can be written as $c(a_{\langle r,p \rangle}) = \delta \times k + m$, where $k = \lfloor \frac{c(a_{\langle r,p \rangle})}{\delta} \rfloor$. Therefore, $m = c(a_{\langle r,p \rangle}) - \delta \times k$. The transition function for the abstract MDP can be defined as,

$$\mathcal{T}(\mathcal{S}_t, a_{\langle r,p \rangle}, \mathcal{S}_{t+1}) = \begin{cases} \frac{m}{\delta} & \text{if } \mathcal{B}^U_{t+1} = \mathcal{B}^U_t - (k+1) \times \delta \\ & \quad \& \ \mathcal{B}^L_{t+1} = \mathcal{B}^L_t - (k+1) \times \delta \\ & \quad \& \ a_{\langle r,p \rangle} \in \mathcal{K}_{t+1} \ \& \ \mathcal{B}^L_{t+1} > 0 \\ \frac{\delta - m}{\delta} & \text{if } \mathcal{B}^U_{t+1} = \mathcal{B}^U_t - k \times \delta \\ & \quad \& \ \mathcal{B}^L_{t+1} = \mathcal{B}^L_t - k \times \delta \\ & \quad \& \ a_{\langle r,p \rangle} \in \mathcal{K}_{t+1} \ \& \ \mathcal{B}^L_{t+1} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

where the budget intervals of states $\mathcal{S}$ and $\mathcal{S}'$ are respectively defined by $[\mathcal{B}^L_t, \mathcal{B}^U_t]$ and $[\mathcal{B}^L_{t+1}, \mathcal{B}^U_{t+1}]$.

Abstracting the state space in MDP not only beneficial by reducing the number of states, but also creates shared states between the policies learned for the different budget values.

Theorem 2. *(***Property of Abstract MDP)** *In the abstract MDP $\bar{\mathcal{M}}$, if $\delta \geq \frac{\mathbf{B}_U}{|\mathcal{A}|}$, $\forall a$ $0 \leq c(a) \leq \delta$, then for any two initial states $\mathcal{S}_0$ and $\mathcal{S}'_0$ with different budget intervals, $\mathbf{S}(\mathcal{S}_0) \cap \mathbf{S}(\mathcal{S}'_0) \neq \varnothing$, where $\mathbf{S}(\mathcal{S}_0)$ and $\mathbf{S}(\mathcal{S}'_0)$ are respectively the sets of all the states that are reachable from states $\mathcal{S}_0$ and $\mathcal{S}'_0$ in the MDP $\bar{\mathcal{M}}$.*

Using the transition function of abstract MDP $\bar{\mathcal{M}}$, defined in the Equation 3.5, and the assumption that $\forall a \ 0 \leq c(a) \leq \delta$, it can be shown that there is an edge in MDP $\bar{\mathcal{M}}$ between every two states $\mathcal{S} = \langle \mathcal{K}, [\mathcal{B}^L, \mathcal{B}^U] \rangle$ and $\mathcal{S} = \langle \mathcal{K} \cup \{\langle r, p' \rangle\}, [\mathcal{B}^L - \delta, \mathcal{B}^U - \delta] \rangle$ for all actions $a_{\langle r,p' \rangle} \notin \mathcal{K}$ and $\mathcal{B}^L \leq \delta$. By induction, we can then show that every initial state $\mathcal{S} = \langle \{\}, [\mathcal{B}^L, \mathcal{B}^U] \rangle$ is connected to state $\mathcal{S} = \langle \mathbf{K}, [0, \delta] \rangle$, if $|\bar{\mathcal{A}}| \geq \frac{\mathbf{B}_U}{\delta}$.

In addition to abstracting the budget, we also need to reduce the number of polling queries in $\mathbf{K}$ to a smaller set $\bar{\mathbf{K}}$. Different feature/variable extraction techniques that have been studied in the machine learning community [Blum and Langley, 1997] can be used to construct $\bar{\mathbf{K}}$. Among these techniques, we use feature ranking using weights from a linear SVM classifier which has been shown to be an effective approach for feature

selection [Guyon et al., 2002]. The SVM is trained using the training data set and the top K ranked polling queries are chosen to be included in $\bar{\mathbf{K}}$. In our experiments, we drop all polling queries with zero weight and keep the rest as part of $\bar{\mathbf{K}}$.

Given the abstract MDP, we can directly learn a policy using the value iteration algorithm. Instead of calculating the policy as the optimal action to be taken from each state, we save the value of $Q(\mathcal{S}, a_{\langle r,p \rangle})$ for all the states $\mathcal{S}$ and actions $a_{\langle r,p \rangle}$, and calculate the optimal policy during the test time given at each state. This allows us to make sure that the action that we are choosing is always within our remaining budget. For example, during the test time, if we are in a state where the budget interval is $[0, 10]$ but the actual remaining budget is equal to 4, we should eliminate selecting actions that require budgets higher than 4.

Our approach to abstract the MDP and solve it using the value iteration algorithm needs to tune a few parameters such as the value of δ. In addition, even with the abstraction, the size of the state space could still be huge when the ontology is very large. Although in the experimental results, we show that by solving the abstract MDP we could significantly outperform other baseline approaches, in the next section we explain how to approximate the value iteration algorithm using Q-learning with linear function approximation which might be more suitable for applications with a very large ontology.

AskWorld (PQL): Q-Learning with Linear Function Approximation

An alternative approach to solve the MDP is to approximate the policy using the temporal-difference, or TD Q-learning, with function approximation [Sutton and Barto, 1998; Lagoudakis et al., 2003]. In the standard setting of TD Q-learning with linear function approximation, the Q-function is represented as a weighted combination of a set of features as follows,

$$Q_\theta(\mathcal{S}, a) = \sum_i \theta_i \phi_i(\mathcal{S}, a)$$

where $\phi_i(\mathcal{S}, a)$ are the features defined over state $\mathcal{S}$ and action a, and θ_i are the set of weights to learn. To learn the parameters θ_i, we follow the standard setting [Sutton and Barto, 1998; Lagoudakis et al., 2003], where an online algorithm is used to update the values of θ_i and reduce their temporal differences between successive states.

3.3.3 KNOWLEDGE INTEGRATOR

Using the training data Ψ, we first train the predictor $h(x, y, z)$ (e.g., SVM classifier) and then calculate the Q values using either the state abstraction or function approximation techniques. Given a trained predicator h and the Q function, we follow the policy defined by Q starting from state $\langle\{\}, B\rangle$. On reaching the last state, denoted by $\mathcal{S}_n$, the label for x is calculated as:

$$\mathbf{y}^* = \arg\max_{y \in \mathcal{Y}} h(x, y, z_n)$$

3.4 Experimental Results

In this section, we evaluate the following two questions:

- Can the knowledge-on-demand system provide a more accurate response if it were allowed to query *all* of the predicates in the Ontology? (Section 3.4.2)
- Is AskWorld effective in identifying the right resources to query under the test-time runtime constraints? (Section 3.4.3)

We describe the datasets and the setup used to answer the above questions in the subsequent sections.

3.4.1 Datasets & Setup

For the experiments in this section, we use 25 categories (predicates with one argument), randomly chosen from all the categories that are in common between the Freebase [Bollacker et al., 2008a] and NELL [Carlson et al., 2010a] knowledge bases. It is necessary to make sure that the selected categories are part of the ontology of both knowledge bases, since the resources used in our experiments (see below) can be only queried for the predicates that are part of the NELL's ontology. Figure 3.2 shows the list of categories used in our experiments.

TelevisionShow	Athlete	Book	Bridge
PoliticalParty	Disease	Hospital	Magazine
VideoGame	Movie	Mountain	Newspaper
PlaceOfWorship	School	Politician	Restaurant
ShoppingMall	Protein	University	Language
MusicArtist	City	Company	Musician
MedicalProcedure			

Figure 3.2: List of the 25 categories used in our experiments. For each category, we use 200 training and 50 test instances.

For each predicate, all the instances are chosen randomly from the Freebase knowledge base. 200 instances are provided as seed examples to train AskWorld. The seed examples are split into two sets: *training* and *evaluation*. The training set is used to train the SVM classifier and the evaluation set is used to solve the MDP (used as part of the reward function) or to find greedy ordering for the baseline approaches. 50 instances are also randomly chosen as the test data. The test data for each predicate p consists of 25 positive examples (i.e., instances of the predicate p) and 25 negative examples. The negative examples are chosen from the other predicates in the FreeBase ontology. We compare AskWorld to the baseline approaches using the standard performance metrics of precision, recall, and F1 score.

For the resources in $\mathcal{R}$, we use three different knowledge acquisition techniques that are developed as part of the NELL project.

- CMC: The first one is CMC which looks for certain orthographic features in the query entity name. For example, if the query noun phrase ends in the suffix "*-burgh*" (e.g., in *Pittsburgh*), then it is likely to be a location.

- KB: The next resource is the NELL KB itself, which is built using several sub-components. One such sub-component is SEAL which checks for recurring patterns in HTML tables on the Web. For example, if SEAL finds that *London* and *Pittsburgh* are often mentioned in the same column of different tables across the Web and we already know that *London* is a location, then it will infer that *Pittsburgh* is

also a location with high probability. Another sub-component is CPL which learns to associate textual patterns in unstructured text to infer specific categories. For example, if CPL finds that the noun phrase *Pittsburgh* is often expressed in the context of pattern *"lives in __"*, then *Pittsburgh* is likely to be a location.

- CKB: While KB has high precision and low recall, CKB is its noisy version (low precision), but covers many more facts (high recall). Queries against CKB tend to be slower since it has higher coverage.

Please note that the performance of these resources varies significantly across predicates, and thus given an instance, it is not always clear which resource to query and how to combine their result. This is especially important in settings with runtime budget constraints which prohibits exhaustive querying of all the resources.

3.4.2 Does polling KRs for non-query predicates help?

One of the contributions of this chapter is to show that the accuracy of a knowledge-on-demand system is improved by aggregating opinions of different KRs for *all* the predicates in the ontology, compared to when we only poll KRs for the query predicate. For example, to answer a query for target predicate *Mountain*, we may benefit by querying some other predicates in the ontology (e.g. *Food*), since it is highly likely that a query entity is not a *Mountain* if it is also a *Food*.

Figure 3.3 shows the precision-recall curve comparing the setting when only the query predicate (referred to as *Single Predicate*) is allowed compared to the setting when we poll KRs for all predicates in the ontology (*All Predicates*). All results are averaged over all 25 target predicates. The result of the different queries are aggregated using a trained SVM classifier. The result shows that the precision of our system is improved by around 25% absolute when we aggregate opinions of KRs for all the predicates in the ontology compared to the single predicate approach, e.g., the precision is improved from 0.64 to 0.81 at the recall value of 0.75.

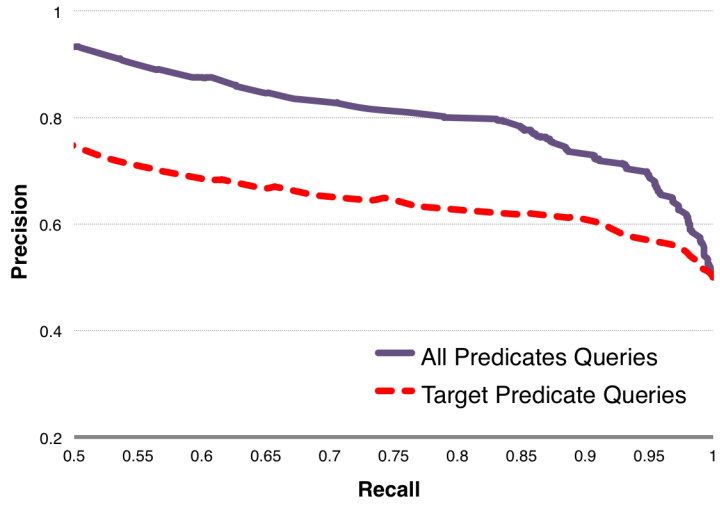


Figure 3.3: Precision-Recall curve comparing performance of AskWorld when it is restricted to issue queries only related to the target predicate (*Target Predicate Queries*) vs. when it is allowed to issue queries for all predicates in the ontology (*All Predicates Queries*). All results are averaged over all 25 target predicates. We observe that allowing more diverse queries, as in the *All Predicates Queries* setting, significantly improves performance.

3.4.3 Are budget-sensitive policies able to select effective polling queries?

Note that the *All Predicates* setting in the previous section is slower than the *Query Predicate* approach since it requires polling KRs for all of the predicates in the ontology. This leaves us with the question of how quickly we can achieve the same result as the *All Predicates* setting while using the minimum budget. To answer this question, we consider the following baselines:

- **Random**: The first baseline that we consider is *Random*, where given a test time budget, we randomly choose which resource and predicate to query until running out of the budget. The values returned for different queries are given as an input to the trained SVM classifier, with missing values in the input of the classifier represented by zeros.

- **Random+**: Random+ is similar to Random, except that missing feature values are represented by prior values calculated during the training (as explained in Section 3.3.1).

- **Single Greedy+**: This method ranks queries in the greedy order based on their information gain. For each predicate in the ontology, the algorithm iterates over all the queries in $\mathbf{K}$, measures the information gain for each single query given all the queries chosen in the previous iterations, and then selects the query with the maximum information gain. The algorithm finds only one ordering for queries in $\mathbf{K}$ for each predicate in the ontology (for each predicate, a separate ordering is found). To measure the information gain for each query, we use trained SVM and measure its accuracy on the evaluation dataset. The missing values in the input of the classifier are represented by the features' prior values. During the query-time, given a new query, the algorithm polls resources by following the greedy ordering until running out of time budget.

- **Multiple Greedy**: The *Multiple Greedy* algorithm is similar to the *Single Greedy+*, except that it iterates over different potential budget values and finds a greedy ordering for every predicate and for every such budget value. Also, instead of training a single SVM and handling the missing values using the features' prior values, *Multiple Greedy* trains a separate SVM for each possible budget value. For example, if the upper bound budget is 10 (integer value), the algorithm finds 10 different greedy orderings for different budget values (1 to 10), and for each one, it trains a separate classifier.

- **Greedy Miser:** *Greedy Miser* is the learning approach presented by [Xu et al., 2012]. We consider Greedy Miser to be representative of the state-of-the-art. Greedy Miser uses step-wise regression [Friedman, 2000] which minimizes a loss function that explicitly trades off the feature cost and the accuracy. The output of the learning algorithm is an additive classifier which is a linear combination of a set of regression trees. At each iteration of the learning, the greedy Classification and Regression Tree (CART) algorithm [Breiman et al., 1984] is used to generate a new tree, which is then used in the additive classifier. We use the Greedy Miser code published by the

authors [1] trained with the same training data as our other algorithms.

Note that GreedyMiser handles the input budget-constraint indirectly using a λ parameter which defines the tradeoff between cost and accuracy. We tuned λ over the range of values suggested in[Xu et al., 2012], and used the optimal value of $\lambda = 0.06$ for the experiments in this section. For other parameters, we use a learning rate of 0.1, depth of 2 for each decision tree (depth of higher than 2 makes GreedyMiser inapplicable for small budget values), squared loss function, and a total of 300 regression trees in the final additive classifier.

- **AskWorld (PQL)**: This is one of the proposed methods. *AskWorld (PQL)* shows the result of the Parametric Q-Learning (PQL) approach (Section 3.3.2) where the features for the linear approximation function are chosen as follows. First, each query in $q \in \mathbf{Q}$ is associated with two boolean features: one which indicates if q is acquired as part of the knowledge of the MDP state, and the other indicates if q is selected as an action from the state. This allows us to represent both arguments of $Q(\mathcal{S}, a)$ function in the linear approximation function. The remaining budget in the state is also represented by one feature. We have tried other different choices such as merging actions and states' features, representing the remaining budget by a set of boolean variables etc. However, the other alternative approaches either decreased or did not change the performance.

- **AskWorld (V*)**: The result for *AskWorld (V*)* is obtained by abstracting MDP using $\delta = 5$ (Section 3.3.1), ordering queries using their information gain, and selecting top $k\%$ of features with non-zero information gain. The value of k depends on the size of the memory and the computation resources available to solve the MDP. In our experiments, we choose $k = 50\%$ which results in approximately 15M states in the MDP. Since AskWorld (V*) is trained only on a subset of queries, we may acquire all of the queries using the learned policy and still have some leftover budget. In this case, we follow the greedy policy and acquire knowledge using queries that have not been already sent.

[1] http://www.cse.wustl.edu/~xuzx/research/code/GreedyMiser.zip

Discussion

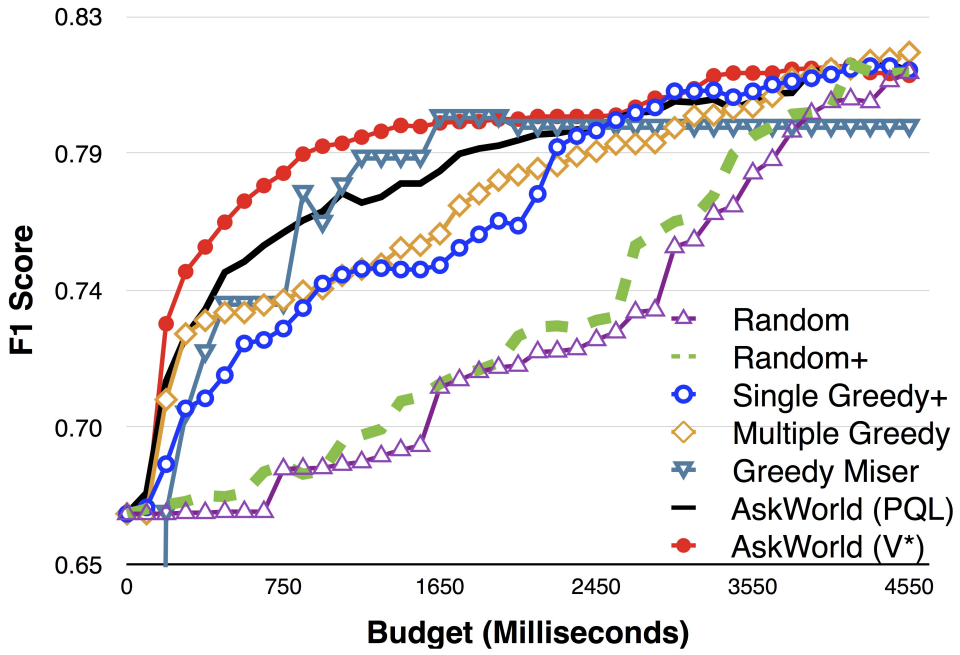


Figure 3.4: F1 scores comparing different systems against varying test-time budgets (in milliseconds). We observe that variants of the proposed approach, AskWorld, outperform all other baselines.

F1 curves comparing these systems are presented in Figure 3.4. In this figure, the x axis is the total test-time budget (in milliseconds) and the y axis is the F1 score (the harmonic mean of precision and recall). Comparing the results for Random and Random+ in Figure 3.4, both algorithms perform equally, except that the F1 score of the Random algorithm is lower at the beginning. The main reason that the Random algorithm is performing poorly at the beginning is that there are many zero elements in the feature vector of the classifier and therefore the constant value in the linear regression function of the SVM plays an important role in biasing the classification result (e.g., at the beginning, when all the features are zeros, only the sign of the constant value in the SVM classifier determines the decision in the classification).

Comparing the results of the two greedy algorithms, we can see that *Multiple Greedy*

outperforms *Single Greedy+* since it learns different greedy ordering and learns a separate SVM classifier for different budget values. Since the *Multiple Greedy* approach requires iterating over the different budget values, its applicability is limited when the upper bound on the test time budget is very large.

The figure also shows the result for the state-of-the-art approach: Greedy Miser [Xu et al., 2012]. Comparing the result of the Greedy Miser with the Single Greedy and Multiple Greedy algorithms, we can see that Greedy Miser performs poorly at the beginning (budget values less than 400), but then outperforms the other algorithms for larger budget values. GreedyMiser uses a different classification technique (cascade of decision trees) and does not achieve the same F1 score as other approaches at higher budget levels.

From Figure 3.4, we observe that both AskWorld(PQL) and AskWorld(V*), our proposed approaches, perform better than all the baselines, including *Single Greedy+*, *Multiple Greedy* and *Greedy Miser*. Note that the *Multiple Greedy* approach requires iterating over all the budget values with one model per budget value, and therefore does not scale as the budget cost increases. In contrast, AskWorld is capable of handling varying test-time budgets (up to a specified upper bound) without retraining and in a *single* model. Comparing the results of Greedy Miser and AskWorld, we observe that AskWorld significantly outperforms Greedy Miser for smaller values of the test budget, and achieves similar results for the larger budget values. For example, for the budget value of 750 Milliseconds, the Greedy Miser algorithm achieves F1 score of 0.74, while the result of PQL and V* are about 0.76 and 0.79, respectively.

In summary, AskWorld, our proposed approach, is able to improve the performance by selecting more informative queries for a given predicate, and without increasing the model footprint.

Examples of Policies Learned by AskWorld

Figure 3.5 shows three examples of policies learned by AskWorld (V*) system for three different predicate instances. AskWorld interactively queries any of the available resources until it exceeds the input budget. At each step, it chooses one of the 25 predicates in Table 3.2 and query from any of the 3 resources (total of 75 = 25 x 3 different queries). At

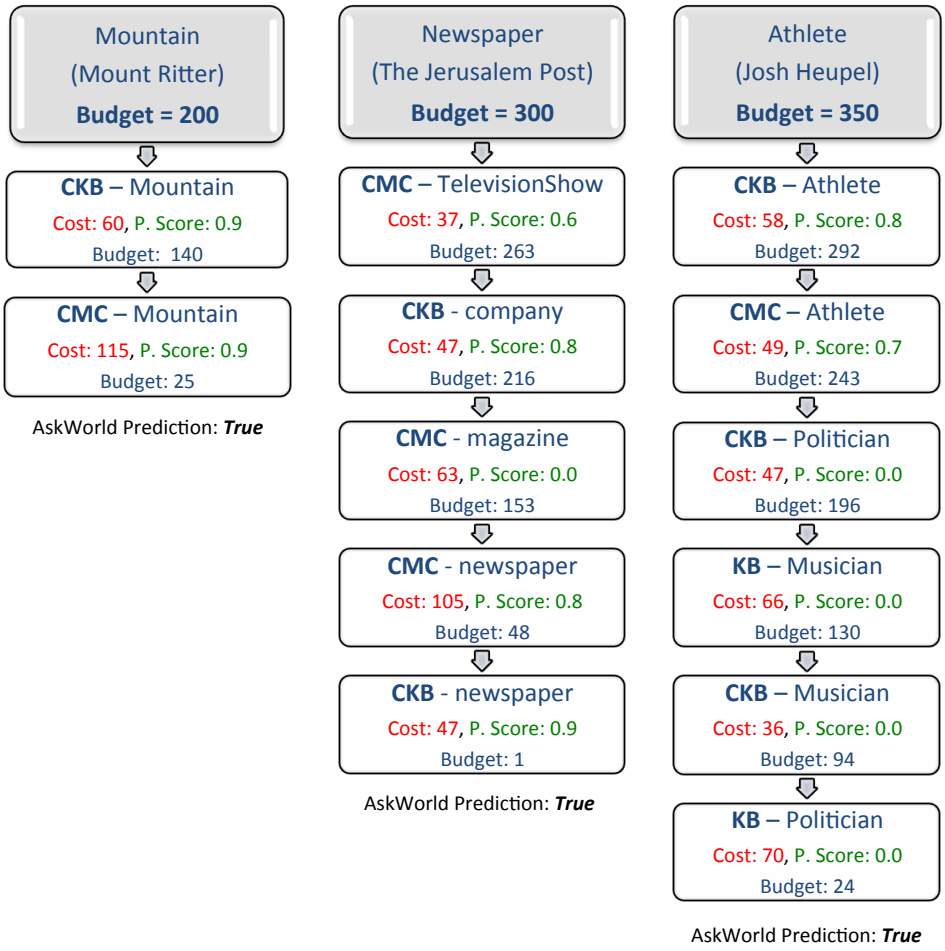


Figure 3.5: Three examples of policies learned by the AskWorld (V*) system. For each example, the input query to the system is first shown (with grey background), followed by the ordered list of queries asked by AskWorld. For each query asked by AskWorld, we show the name of the resource that is queried (in bold), the name of the predicate, the cost of the query (in milliseconds), the prediction score returned by the resource, and the leftover budget. In each case, AskWorld predicts *true*, the correct response.

each step, the figure shows the query that has been asked, and the value returned for each of the queries. For all of these queries, AskWorld correctly returns the *true* label.

For example, for the query *Mountain(Mount Ritter)*, since the budget is small, AskWorld is conservative and only checks if *Mount Ritter* is an instance of *Mountain* using *CKB* and *CMC* resources. For the query *Athlete(Josh Heupel)*, where a larger budget is available, AskWorld first gets some evidence about the input instance being an *Athlete* by querying *CKB* and *CMC* resources. Both CKB and CMC respectively return the prediction score of 0.8 and 0.7 (the probability of the input predicate instance being correct). Since there are still some budgets available, AskWorld explores other categories that are potentially mutual-exclusive with predicate *Athlete* by getting prediction scores for predicates *Politician* and *Musician* (e.g., a musician usually is not an athlete). All the resources return the prediction score of 0.0 for *Josh Heupel* to be an instance of either *Politician* or *Musician*, which increases the confidence of AskWorld in correctly predicting that *Josh Heupel* is indeed an *Athlete*.

3.5 Summary

In this chapter, we presented AskWorld, a novel system which is capable of providing knowledge-on-demand. We showed that the accuracy of knowledge acquisition improves when the system is allowed to issue polling queries corresponding to non-query predicates in the ontology. Even though this relaxation results in an explosion of polling possibilities, AskWorld is able to select the most informative ones within runtime budget constraints. To the best of our knowledge, AskWorld is the first knowledge-on-demand system of its kind which is capable of handling varying query-time budgets without model retraining. Through extensive experiments on real world datasets, we demonstrated AskWorlds capability of selecting most-informative queries within query-time runtime constraints, resulting in improved performance while achieving reduced model footprint.

Chapter 4

Measuring Credibility of Sources and Extracting Reasons

In this chapter, we present ClaimEval, a novel and integrated approach which given a set of claims to validate, extracts a set of pro and con arguments from the Web using OpenEval, and jointly estimates credibility of sources and correctness of claims. ClaimEval uses Probabilistic Soft Logic (PSL), resulting in a flexible and principled framework which makes it easy to state and incorporate different forms of prior-knowledge.

We begin in Section 4.1 by motivating the credibility assessment problem. In Section 4.2, we explain the details of our ClaimEval technique. In this section, we first explain how to use OpenEval for automatically building a Credibility Assessment (CA) graph. We then motivate and present the details of the assumptions that we have used throughout the rest of this chapter for assessing the credibility of sources and validating the truth of the claims (i.e., prior knowledge). The details of our approach for using Probabilistic Soft Logic (PSL) to formalize representation of the prior knowledge, and to find the truth of claims and credibility score of sources are then discussed. The experimental results and comparison with baseline approaches are presented in Section 4.3.

4.1 Introduction

The Internet is a source of a vast amount of information, and any individual who has access to the Internet can publish materials that can be reached by millions of people. The Web search engines, such as Google, Bing, etc., that operate over millions of documents have made this information readily available to everyone. However, despite the ease of access to enormous amounts of material on the Web, it is now much harder for an average Internet user to assess the credibility of a particular source of information. Assessing the credibility of an information source is especially important when a user is seeking to validate correctness of a given *claim*.

To make it possible to readily evaluate *factoid* claims (e.g., *"Paris is the capital of France"*), several large knowledge bases (KBs), such as Freebase, Yago, Google Knowledge Graph, etc., have been developed in recent years. In spite of this democratization of information, evaluating the correctness of *non-factoid* claims (e.g., *"Turkey meat is healthy"*), is still an open challenge. This is particularly challenging since two different webpages may contain conflicting evidences even related to a single claim. For example, while an animal rights website might not support meat-eating and thus classify turkey meat as unhealthy, the website of a grocery store might claim otherwise. A scientific paper focusing on this question might provide the most authoritative answer. One would ideally want to trust evidences contained in the credible source (the scientific paper) and ignore the other two. Hence, given a set of claims, one needs to identify relevant sources on the Web, extract supporting and contradictory evidences from those sources, estimate source credibility, and finally aggregate all of the information to evaluate the given set of claims. This is a tedious and time consuming process, and current Web search engines only address the first aspect of this bigger problem, viz., identifying relevant sources. Thus, there is a growing need for an *integrated* approach for automatic extraction of relevant evidences and sources, estimation of information source credibility and utilizing those credibility estimates in claim evaluation.

Estimating information source credibility may also be subjective [Bhattacharya et al., 1998; Gambetta, 1990]. In other words, we may have to take user preferences and context into account when estimating source credibility and evaluating claims. This can be

illustrated through an example shown in Figure 4.1. In this example, there are two claims: *"Turkey is healthy"* (C1) and *"Chicken is healthy"* (C2). These claims are either supported (*SupportsClaim*) or refuted (*DoesNotSupportClaim*) by evidences originating from webpages of two domains, *peta.org*, website of an animal rights activists' group, and *whfoods.com*, a non-profit promoting healthy food. The user has also indicated that she believes C1 is in fact true (shown by a green double-line rectangle). Given this initial information, we would like to evaluate whether claim C2 is true or false. In this case, we find that all the evidences originating from the domain *peta.org* are in contradiction with the user's assessment of claim C1. Hence, we would like to decrease the credibility of *peta.org* and reduce the influence of all evidences originating from it while evaluating C2. We note that the evidences from domain *whfoods.com* is in agreement with the user's assessment of C1, and hence we should increase its credibility. We note that these credibility adjustments are claim (and user) specific. If instead of claims involving healthy meat options, they were focused on animal cruelty, then *peta.org* might have been a very credible source.

The above example shows that the correctness estimates of claims in the Credibility Assessment (CA) graph and the credibility of non-claim nodes can be calculated by propagating any available information about the correctness of claims or the credibility information of other nodes over the CA graph. Our intuition is that humans naturally do this type of credibility and correctness propagation based on *prior-knowledge* using a set of rules. The prior-knowledge specifies how the credibility inference should be performed across the CA graph. Ideally, in order to automatically calculate the credibility of a set of claims and sources, a credibility assessment approach should be able to take the prior-knowledge as an input and incorporate it in its credibility calculation.

In this chapter, we introduce a novel and fully-automated technique, called ClaimEval, which validates the truth of a set of claims, by automatically crawling the relevant information from the Web using the OpenEval approach (introduced in Chapter 2), calculating the credibility of sources, and incorporating the calculated credibility scores to validate the truth of the given claims. When presented with a claim to verify, ClaimEval first uses OpenEval to search the Web for different evidences (i.e. context-based instances) that are in favor or against the input query claim. OpenEval automatically extracts evidences using

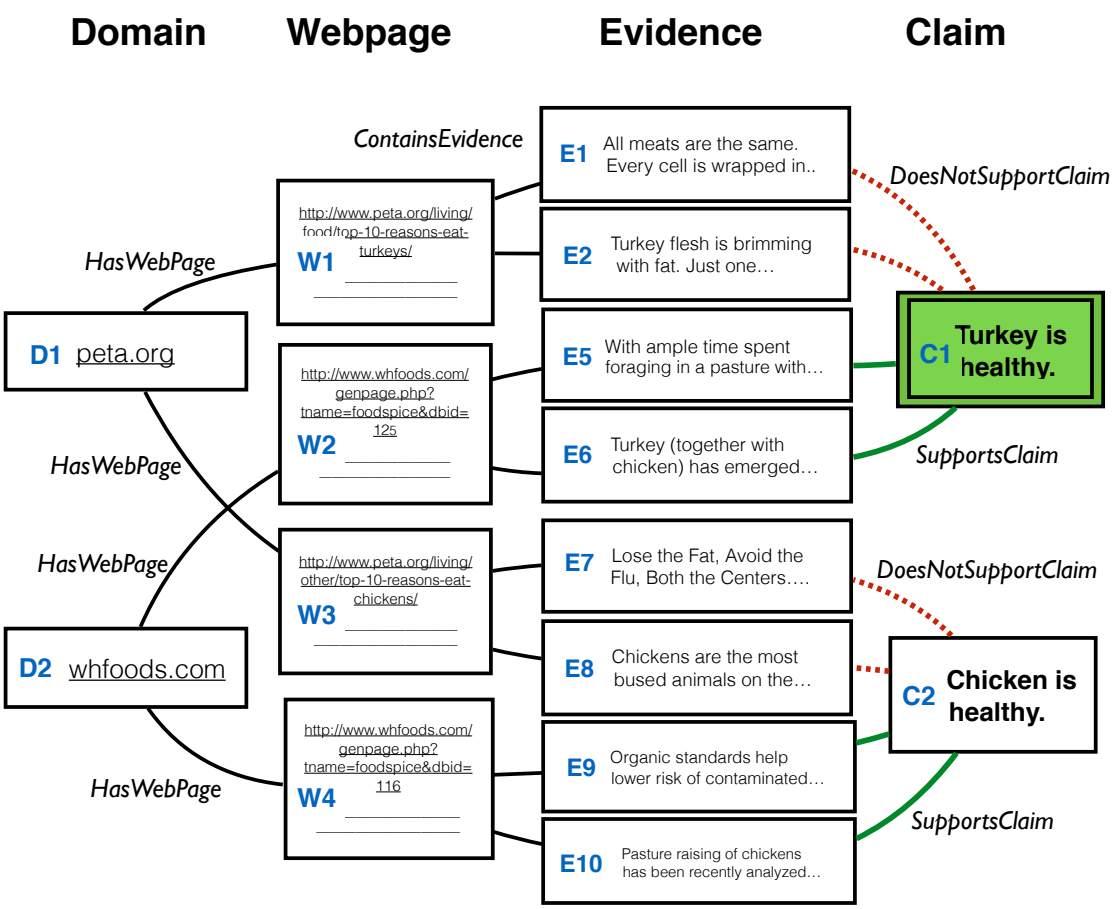


Figure 4.1: The Credibility Assessment (CA) Graph is constructed to evaluate C2, while the user has already specified the other claim C1 to be true (green concentric rectangle). In addition to the claims, the CA graph also consists of nodes corresponding to *Domains*, *Webpages* and *Evidences*. While some evidences are supportive of the claims (solid green lines), others are not (red dotted line) (see Section 4.2.1 for details). ClaimEval, the proposed system, estimates the credibility of the domains based on the assessment available on claim C1, and combines that with the level of support from evidences originating from those domains to evaluate claim C2 (see Section 4.2.2 for more details).

both structured and unstructured information on the Web, and measures the sentiment of the extracted evidences (determining if they are against or in favor of a claim) using the

trained classifiers. Given these extracted evidences, ClaimEval then builds the CA graph using the retrieved information. Given the constructed CA graph, ClaimEval iteratively flows the information from the sources to the claims, and then back to the sources, in order to find the credibility of the sources and the truth of the claims.

To represent the prior knowledge on how the credibility of sources influence claims and vice versa, we first define a set of simple rules that explicitly define how the information in the graph should flow between the nodes, e.g. if an evidence is coming from a credible website and the evidence is supporting a claim to be true, then the claim is true. Each node in the graph is represented by a ground atom, which takes a soft truth value in the interval $[0, 1]$. We show how to represent these rules in the form of a set of first-order logical rules. Using Probabilistic Soft Logic (PSL) [Broecheler et al., 2010], that is a framework for probabilistic reasoning in relational domains, we present a novel credibility approach that takes these logical rules and the CA graph, builds a joint probabilistic model over all the nodes in the graph, and assigns values to the different nodes by solving a continuous optimization problem. The values are assigned to the nodes in the graph so as to have maximum consistency with the set of defined rules. Our technique is efficient and its time complexity is polynomial in terms of the number of nodes in the graph.

4.2 Our Approach: ClaimEval

ClaimEval performs joint estimation of source credibility and claim evaluation in one single model using Probabilistic Similarity Logic (PSL). Algorithm 7 summarizes the process that ClaimEval follows for evaluating a claim. ClaimEval consists of the following two steps: (i) **Credibility Assessment(CA) Graph Construction** (Section 4.2.1 - Line 1 in Algorithm 7), and (ii) **Joint Source Credibility Estimation and Claim Evaluation** (Section 4.2.2 – Lines 7-8 in Algorithm 7).

In the following two sections, we describe each one of these two steps in greater detail.

Algorithm 7 ClaimEval - Evaluating the correctness of a alaim

Require: $\langle U, L, r, K \rangle$ /* U is a set of unlabeled claims that should be evaluated, L is a set of labeled claims (label is either true or false), r is the category that U and L belong to, and R is a set of first-order logic rules defining the prior credibility knowledge. */

Ensure: $\langle$label (True or False), confidence$\rangle$ for claims in U

1: *CAG* $\leftarrow$ Construct the CA graph by calling **BuildCAGraph (**U, L, r**)** //Algorithm 8
2: $\hat{R} \leftarrow$ Relax logical operators in R using *Luka-siewicz* real-values operators
3: $P \leftarrow$ Convert $\hat{R}$ and *CAG* to an optimization problem using Probabilistic Soft Logic (PSL)
4: $[\langle$labels,confidence values$\rangle] \leftarrow$ Solve the optimization problem P and find the label (True or False) and confidence value for each unlabeled claim in U
5: **return** $[\langle$labels,confidence values$\rangle]$

4.2.1 Credibility Assessment (CA) Graph Construction

Let us consider the two claims: *"Turkey is healthy"* (C1) and *"Chicken is healthy"* (C2) as shown in Figure 4.1. In practice, in order to evaluate these claims, one would probably first try to identify sources (e.g., webpages) which are likely to contain evidence either in favor or against such claims. An evidence may be considered as a text snippet (e.g., a sentence or paragraph in a webpage) expressing an opinion about the claim in a webpage. Overall, a *claim* is substantiated by one or more *evidences* which are contained in *webpages*, with webpages in turn contained within *domains*. This is combined in a multi-relational graph that is called a Credibility Assessment (CA) graph. A CA graph has four types of nodes:

- *Domain*: A domain name is an identification string that defines a realm of administrative control within the Internet. Domain names are used in URLs to identify particular webpages.
- *Webpage*: A webpage is a web document.
- *Evidence*: An evidence is a small section of text in a webpage.
- *Claim*: A claim is a category instance, of which we may or may not know the truth value.

Nodes are connected by the following five types of edges:

- *HasWebPage(Domain, WebPage)*: Connects a webpage of a specific domain.
- *ContainsEvidence(WebPage, Evidence)*: Connects an evidence that is extracted from a particular webpage.
- *DoesNotSupportClaim(Evidence, Claim)*: Connects an evidence to a claim, where the evidence does not support the claim.
- *SupportsClaim(Evidence, Claim)*: Connects an evidence to a claim, where the evidence supports the claim.

For example, in Figure 4.1, evidence node *"All meats are the same..."* (E1) *DoesNotSupportClaim* C1. Evidence E1 is found in the webpage represented by webpage node W1. This particular webpage is from the *peta.org* domain which is represented by node D1 in the CA graph.

The input of ClaimEval is similar to the input of OpenEval system, where a set of categories of claims, and a set of *true* and *false* claims for each category (i.e., category instances), are provided as input. For example, *healthy food* is an example of a category, and *apple, broccoli* and *mayonnaise, soda* are, respectively, *true* and *false* claims provided for this category. In addition to the labeled instances which are used for training, we assume that a set of unlabeled instances are also provided, for which ClaimEval calculates a credibility score. Note that in the rest of this chapter, each claim is simplified by only showing the category instance, for example, a claim such "apple is a healthy food" is shown by "apple" as an instance of "healthy food" category. Our goal is to classify the set of unlabeled claims to either *true* or *false* classes, with a confidence value for the assigned label.

The algorithm for building the CA graph is shown in Algorithm 8. To build the CA graph, ClaimEval first uses the OpenEval system to extract a set of evidences (i.e., context-based instances) and learn an *evidence classifier* for each category of claims, which is later used to classify evidences for a given claim to either *pro* or *con* classes. As we discussed in Chapter 2, to build the training data for the classifier, OpenEval first searches each claim on the Web, and extracts a set of evidences (Context-Based Instances) from the returned webpages. Using the extracted training data, it then learns an *evidence classifier*

that assigns each extracted evidence to either the *pro* or the *con* class. Given the extracted evidences, ClaimEval automatically constructs the CA graph (similar to Figure 4.1) for all the claims.

Algorithm 8 ClaimEval - Building the Credibility Assessment graph

Require: $\langle U, L, r \rangle$ /* U is a set of unlabeled claims, L is a set of labeled claims (label is either true or false), and r is the category that U and L belong to. */

Ensure: *CAG* $\leftarrow$ A Credibility Assessment Graph

1: **Function: BuildCAGraph** ($\langle U, L, r \rangle$)
2: *CAG* $\leftarrow$ Initiate empty layered-graph *CAG* with four layers: *Domains, Webpages, Evidences, Claims.*
3: $R \leftarrow$ List of categories of claims in $(U \cup L)$
4: **for** each category $r \in R$ **do**
5: $T_r^+ = T_r^- = \{\}$. %Initiate empty positive and negative training data set for category r.
6: $C_r \leftarrow$ Claims of category r in $(U \cup L)$
7: **for** each claim $c \in C_r$ **do**
8: $E \leftarrow$ Extract evidences (CBIs) for claim c using the OpenEval approach
9: **if** c is labeled **then**
10: If c is labeled *true*, then add set E to T_r^+, otherwise add it to T_r^-.
11: **end if**
12: **for** each evidence $e \in E$ **do**
13: Add e as a node in the evidence layer of *CAG*.
14: Add claim c to the claim layer, connect e to c.
15: Add the webpage and the domain, from which e is extracted, to webpage and domain layers in *CAG*, respectively. Add an edge between them.
16: **end for**
17: **end for**
18: Convert each element of T_r^+ and T_r^- to a feature vector
19: $h_r \leftarrow$ Train classifier h_r using training data T_r^+ and T_r^-
20: Mark each edge that connects evidence e to claim c as *pro* if $h_r(e) \geq 0$, otherwise mark it as *con*.
21: For all the labeled claims, label the corresponding node in the *CAG*.
22: **end for**
23: **return** *CAG*

After training the evidence classifier, ClaimEval iterates over all the labeled and unlabeled claims, similar to the training process, and extracts a set of evidences for each claim.

Using the trained evidence classifier, each of the extracted evidences is then assigned to either the *pro* or the *con* classes.

The evidence layer in the CA graph is built using all the evidences extracted from the Web. Each evidence is connected to the claim for which it is extracted. The edge that connects an evidence to a claim, is classified as *SupportsClaim* (solid green line in Figure 4.1), if the evidence is supporting the claim, and otherwise is classified as *DoesNotSupportClaim* (red dotted line).

The webpage layer is constructed from the webpages that are used by the OpenEval. Each webpage is connected to the set of evidences extracted from the webpage. For each webpage, we extract the domain name (e.g., *whfoods.org*) and create a corresponding node in the domain layer. Each domain node is connected to its webpages in the webpage layer.

Initially, all the nodes in the graph do not have any value, except for the nodes whose true labels (i.e., user assessments) are known. Each node in the CA graph takes a value in the range [0,1]. The semantic of the values assigned to the nodes varies across the layers in the graph. For example, the value of a node in the webpage layer is interpreted as the degree of the credibility of the webpage, and the value of a node in the claim layer is interpreted as the confidence in the truth of a claim. Edges in the graph can be seen as transforming the meaning of *values* of nodes between the layers.

Given the CA graph, we can iteratively calculate the credibility of each source by using the claims that it asserts, and simultaneously calculate the truth of each claim given the credibility score of the sources that are asserting it. For example, using a simple voting algorithm, we can calculate the probability of a claim to be either *true* or *false*, by simply dividing the total number of evidences that are in favor of a claim (*pro* evidences), by the the total number of evidences that are connected to a claim.

If we treated all evidences as equally credible, then using a simple voting rule, we can estimate the likelihood of claim C1 in Figure 4.1 being true is $\frac{2}{4} = 0.50$ (only two of the four evidences are supporting it). Similarly, claim C2 being true has a likelihood of 0.5. However, we can propagate the information about correctness of claim C1 over the CA graph to revise the correctness estimates claim C2. Our intuition is that humans naturally do this type of credibility and correctness propagation based on *prior knowledge* using a

set of rules. We explain these rules in the next section.

4.2.2 Joint Source Credibility Estimation & Claim Evaluation

When computing the credibility scores in the CA graph, we propagate the credibility and the correctness information across the different layers based on some *prior knowledge*, which is defined as a set of rules. We first list a set of such rules that specify how the credibility inference should be performed across the CA graph. We later explain how these rules are incorporated in our credibility assessment model.

Prior Knowledge for Credibility Assessment

- **Evidence** $\Rightarrow$ **Claim**: Inferring correctness of a claim based on the credibility of an evidence:
  - EC1: Evidence is *credible* & evidence *supports* claim $\Rightarrow$ claim is *true*.
  - EC2: Evidence is *credible* & evidence *doesn't support* claim $\Rightarrow$ claim is *false*.
  - EC3: Evidence is not *credible*, then the evidence has no effect on the correctness of the claim.
- **Claim** $\Rightarrow$ **Evidence**: Inferring credibility of an evidence based on the correctness of a claim:
  - CE1: Claim is *true* & evidence *supports* claim $\Rightarrow$ evidence is *credible*.
  - CE2: Claim is *true* & evidence *doesn't support* claim $\Rightarrow$ evidence is not *credible*.
  - CE3: Claim is *false* & evidence *supports* claim $\Rightarrow$ evidence is not *credible*.
  - CE4: Claim is *false* & evidence *doesn't support* claim $\Rightarrow$ evidence is *credible*.
- **Webpage** $\Leftrightarrow$ **Evidence**: Inferring credibility of an evidence based on the credibility of a webpage, and vice versa:
  - WE1: Webpage is *credible* $\Leftrightarrow$ evidence is *credible*.

- WE2: Webpage is not *credible* $\Leftrightarrow$ evidence is not *credible*.

- **Domain $\Leftrightarrow$ Webpage**: Inferring credibility of a webpage from the credibility of a domain, and vice versa:

  - DW1: Domain is *credible* $\Leftrightarrow$ webpage is *credible*.
  - DW2: Domain is not *credible* $\Leftrightarrow$ webpage is not *credible*.

Encoding Prior Knowledge using First-Order Logic

We use first-order logic (FOL) to formally define a set of rules, based on the prior knowledge that we defined in Section 4.2.2. Each layer of the CA graph is represented by a logical *predicate*, and each node in the graph is an instance of the predicate. For example, the predicate *Domain(x)* is used to define different nodes/instances in the first layer of the graph, where *peta.org* is an instance of this predicate, represented by *Domain(peta.org)*. In this case, predicate instance *Domain(peta.org)* has value 1, since *peta.org* is a domain in the *Domains* layer, otherwise it would have taken the value 0. Similarly, predicates *Webpage(x), Evidence(x),* and *Claim(x)* are defined to represent nodes in the other layers of the graph.

In addition to the predicates that represent different types of the nodes in the graph, we use the following predicates to define edges: *HasWebpage(Domain, Webpage)*, *ContainsEvidence(Webpage, Evidence)*, *SupportsClaim(Evidence, Claim)*, *DoesNotSupportClaim(Evidence, Claim)*. The edges that connect *Evidences* to *Claims* are assigned with either the *pro* or the *con* label, where are represented using the predicates *SupportsClaim(e,c)* and *DoesNotSupportClaim(e,c)*, respectively. If an evidence *e* supports a claim *c*, then the predicate *SupportsClaim(e,c) = 1*, and if it does not support a claim then *DoesNotSupportClaim(e,c)=1*. In all other cases, including when there is no edge between node *e* and *c*, the values of *SupportsClaim* and *DoesNotSupportClaim* are equal to zero. To compute the value of these two predicates, we give evidence *e* to the classifier that is trained for the category that claim *c* belongs to, and assign value 1 to *SupportsClaim(e,c)* if *e* is classified as a *pro* argument, assign a value of 1 to predicate *DoesNotSupportClaim(e,c)* if *e* is classified as a *con* argument, and 0 otherwise.

Table 4.1 shows the set of rules that define the flow of the information between *Evidences* and *Claims* layers. Similarly, Tables 4.2 and 4.3 present the rules that define the information flow between *Domains*, *Webpages*, and *Evidences* layers.

#	**Flow**	**Rule**
EC1	Evidence ⇒ Claim	Evidence(e)∧Claim(c)∧Credible(e) ∧SupportsClaim(e,c) ⇒Correct(c)
EC2	Evidence ⇒ Claim	Evidence(e)∧Claim(c)∧Credible(e) ∧DoesNotSupportClaim(e,c) ⇒ ¬Correct(c)
CE1	Claim ⇒ Evidence	Evidence(e)∧Claim(c)∧Correct(c)∧SupportsClaim(e,c) ⇒Credible(e)
CE2	Claim ⇒ Evidence	Evidence(e)∧Claim(c)∧Correct(c) ∧DoesNotSupportsClaim(e,c) ⇒ ¬Credible(e)
CE3	Claim⇒ Evidence	Evidence(e)∧Claim(c)∧¬Correct(c) ∧SupportsClaim(e,c) ⇒ ¬Credible(e)
CE4	Claim⇒ Evidence	Evidence(e)∧Claim(c)∧¬Correct(c) ∧DoesNotSupportClaim(e,c) ⇒Credible(e)

Table 4.1: First-order logic rules that define how the information flows between an evidence and a claim.

Table 4.1 contains two sets of rules. Rows 1-2 define the flow of information from the *Evidences* layer to the *Claims* layer, and rows 3-6 define the information flow from the *Claims* to the *Evidences* layer. We explain the first two rules in detail, and the rest follows similarly. The first two rules define the flow of information from evidences to the claims, where e is an evidence, c is a claim, and e is connected to c. In the case that the evidence is credible (*Credible(e)* has value 'true' or 1.0), depending on whether e is in *favor* (i.e., *SupportsClaim(e,c)=1*) or against (i.e., *DoesNotSupportClaim(e,c)*=1) the claim, c is labeled true (i.e., *Correct(c)=1*) or false (i.e., *Correct(c)=0*). Note that if the evidence e is not credible, then whether the evidence is a *pro* or *con* argument has no effect on the value of the claim c. This is since a claim made by an evidence that is not credible

can not be trusted. This case can be represented by the following rule:

$$Evidence(e) \wedge Claim(c) \wedge \neg Credible(e) \wedge SupportsClaim(e, c) \Rightarrow Correct(c) \vee \neg\ Correct(c) \tag{4.1}$$

Since the right side of the Equation 4.1 is always equal to one, we drop it from the list of rules in Table 4.1.

Similar to Table 4.1, Tables 4.2 and 4.3 define the flow of information for *Evidences*, *Sources*, and *Domains* layers. These rules are similar to the rules used in SSL semi-supervised algorithms, that make the values of neighboring nodes in a graph similar. Note that some of the rules in these tables are simplified through the use of the following logical rule: *if* $a \Rightarrow b$, *then* $\neg b \Rightarrow \neg a$.

#	Flow	Rule
WE1	Source⇒ Evidence	Source(s)∧Evidence(e)∧ContainsEvidence(e,c) ∧Credible(s) ⇒Credible(e)
WE2	Evidence ⇒Source	Source(s)∧Evidence(e)∧ContainsEvidence(e,c) ∧Credible(e) ⇒Credible(s)

Table 4.2: First-order logic rules that define how information flows between a source and an evidence.

#	Flow	Rule
DW1	Domain⇒ Source	Domain(d)∧Source(s)∧HasWebpage(d,s) ∧Credible(d) ⇒Credible(s)
DW2	Source ⇒Domain	Domain(d)∧Source(s)∧HasWebpage(d,s) ∧Credible(s) ⇒Credible(d)

Table 4.3: First-order logic rules that define how information flows between a domain and a source.

Although the logical rules precisely define the flow of information in the graph, their applicability is limited for real world examples, as the nodes in the graph can only take

values 1 (*true*) or 0 (*false*). For example, assume that a claim *c* is supported by two evidences *a* and *b*, both of which are credible. If *a* is in favor of *c* and *b* is against *c*, then we can neither say that *c* is *true* nor that *c* is *false*. Ideally, the value of claim *c* is between 0 and 1 depending on the relative credibility of evidences *a* and *b*. In the next section, we explain how to address this issue by relaxing the binary value constraint and allowing each node to take any value in the interval $[0, 1]$ using Probabilistic Soft Logic (PSL) [Broecheler et al., 2010; Kimmig et al., 2012].

Going Beyond Binary Logical Operators using Probabilistic Soft Logic (PSL)

Probabilistic Soft Logic (PSL) is a general purpose logical system that uses First-Order Logic (FOL) as its underlying logical language. We provide a brief overview of the PSL technique briefly in this section; for a more detailed exposition of PSL, we refer the reader to [Bach et al., 2012; Broecheler et al., 2010; Kimmig et al., 2012].

In PSL, the value of each ground atom is relaxed to take a soft truth-value in the interval [0,1], where 0 is interpreted as absolute false and 1.0 as absolute true. The soft-truth assignment allows us to define the degree of correctness and credibility for the nodes in the graph. For example, if we have two sources s_1 and s_2, where *Credible*$(s_1) >$ *Credible*(s_2), then we can infer that s_1 is more credible compared to s_2.

To handle the continuous truth values in the variables, PSL relaxes conjunction, disjunction, and negation logical operators by using *Luka-siewicz* real-valued logic operators, which are defined as follows:

$$\begin{aligned} a \,\hat{\wedge}\, b &= \max\{0, a + b - 1\} \\ a \,\hat{\vee}\, b &= \max\{1, a + b\} \\ \hat{\neg}\, a &= 1 - a \end{aligned}$$

Lukasiewicz t-norm operators, compared to other non-classical logic operators such as *product t-norm*, are suitable for our credibility assessment model since they linearly combine the values that they take. For example, consider this rule:

$$\textit{Credible(e)} \,\hat{\wedge}\, \textit{SupportsClaim(e,c)} \Rightarrow \textit{Correct(c)}$$

Using the *Lukasiewicz* disjunction operation, we can write the body of this rule as:

$$\max \{0, \textit{Credible(e)} + \textit{SupportsClaim(e, c)} - 1\}$$

which evaluates to *true* when the resulting value is greater than a certain threshold, say 0.5. We can roughly interpret it as: claim c is *correct* only when e is credible and supports claim c, each at least by a degree of 0.5 (sum should be greater than 1.5).

Given a set of rules, we first instantiate all the variables in the rules with respect to the CA graph that is constructed. Given these instantiated rules, our goal is to find values of the different nodes in the graph such that the total number of satisfied rules is maximized. For solving this optimization problem, PSL defines the satisfaction distance for each rule in the domain. Given a ground rule r, such as $r := P \Rightarrow Q$, r is satisfied if and only if $\hat{V}(Q) \geq \hat{V}(P)$, where $\hat{V}(X)$ is defined as the value of the ground logical expression X using *Lukasiewicz* operators (i.e., Q is at least as truthful as P). The rule's *distance to satisfaction* (denoted by $d(r)$) measures the satisfaction degree of each rule $r := P \Rightarrow Q$:

$$d(r) = \max\{0, \hat{V}(P) - \hat{V}(Q)\} \tag{4.2}$$

Given the distance function in Equation 4.2 and a set of ground rules, PSL finds values for all the ground predicate instances in order to minimize the total satisfaction degrees of all the ground rules. To do this, assume that I is the assignment of values to predicate instances, and $d_I(r)$ is the satisfaction degree of rule r given assignment I. Thus, given the set of ground rules R, the optimal assignment I^* may be obtained as follows,

$$I^* \leftarrow \arg\max_I \frac{1}{Z} \exp[-\sum_{r \in R} \lambda_r (d_I(r))^2] \tag{4.3}$$

where Z is the normalization factor, and λ_r is the weight for each ground rule r. The Most Probable Explanation (MPE) inference algorithm may be used to optimize Equation 4.3.

When optimizing Equation 4.3, we allow only the values of *Credible* and *Correct* predicate instances to change, and the values of the rest of predicate instances remain fixed. In order to make sure that the labeled claims will preserve their values during the optimization (i.e., constraints), we define the following rules:

$$\forall \text{ labeled claims } c,\ \textit{PosClaim}(c) \rightarrow\ \textit{Correct}(c) \tag{4.4}$$

$$\forall \text{ labeled claims } c,\ \textit{NegClaim}(c) \rightarrow \neg\ \textit{Correct}(c) \tag{4.5}$$

Predicates *PosClaims* and *NegClaims* respectively, define the positive and negative set of labeled claims.

4.2.3 Learning Rules Weights

The rules that are defined for trust in Tables 4.1-4.3 and Equations 4.4 and 4.5 have a different degree of importance. For example, rules in Table 4.3 are generally correct. However, different websites that are associated with a domain may have different degrees of trustworthiness. For example, different webpages in the Elsevier journal website may have different qualities (papers with different number of citations), or blogs that are hosted under the same domain name may have a different degree of trustworthy (e.g., a blog of a University professor compared to a blog of a student). Furthermore, rules that are defined in Equations 4.4 and 4.5 should have higher weight compared to the other rules defined as part of the prior knowledge, in order to make sure that during the optimization, the correct labels are assigned to the labeled data.

The PSL framework gives us the flexibility to assign different weights to rules in the graph, by using the Maximum Likelihood Estimation (MLE) [Broecheler et al., 2010]. We use the same training data, which is used to train the classifer, in order to learn the weights for the rules.

4.2.4 Examples

Figure 4.3 shows five different examples of CA graph with credibility scores assigned to each node of the graph. Each graph is slightly modified to show how the credibility scores vary for different scenarios. Each graph contains two claims (nodes 8 and 9), where claim 8 is labeled as *correct* in our training data. Using the structure of the graph and the labels provided for different nodes, our goal is to infer the label of claim 9 (wether it is *true* or *false*). We explain the rational behind assigning scores to different graphs below.

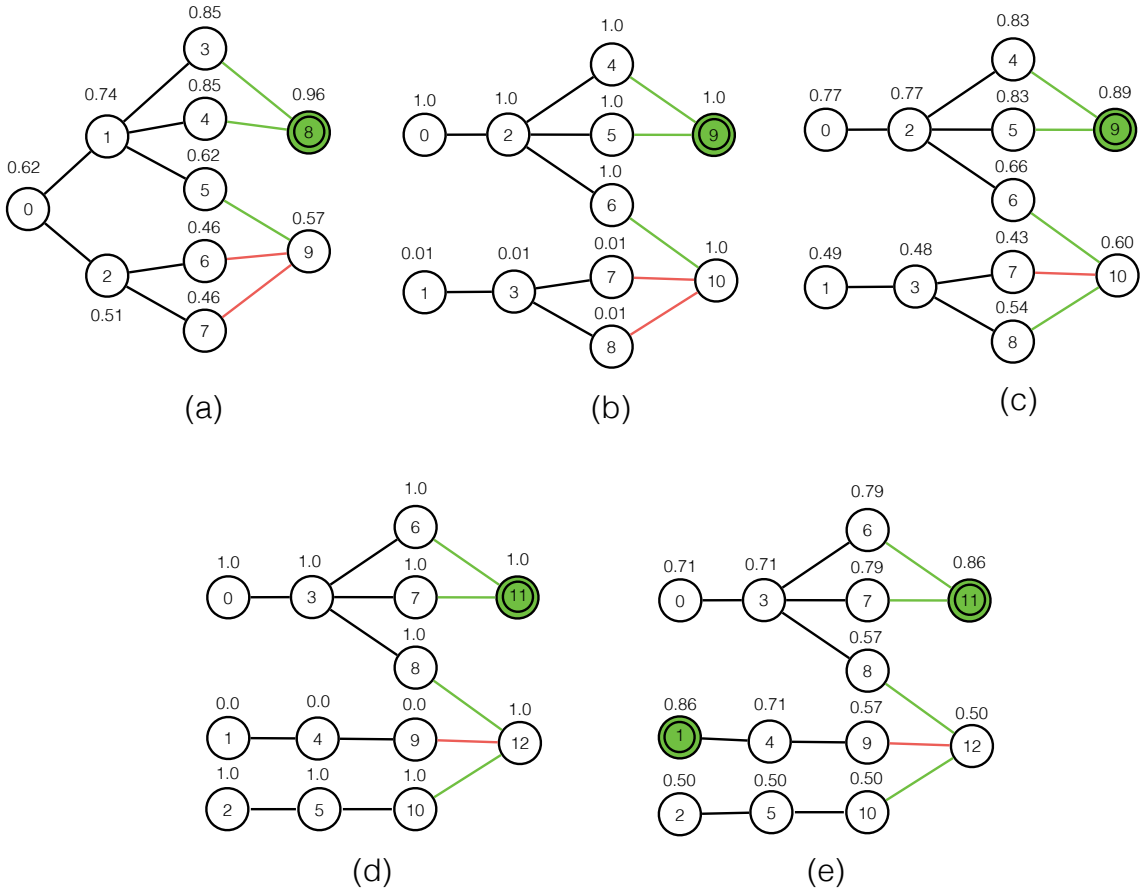


Figure 4.2: Five examples of the Credibility Assessment Graph, with the credibility score assigned to each node in the graph. Similar to Figure 4.1, the graphs' layers from the left are: *Domains*, *Webpages*, *Evidences*, and *Claims*. A redline connecting an evidence to a claim is an indicator of a *con* argument, while a *green* line is an indicator of a pro argument. White nodes are the nodes where the credibility value is not known, and a node with the green color means that it is credible. A double-lined circle indicates the training data (i.e., nodes which we know the truth value).

1. **Graph (a)**: Since nodes 3 and 4 (evidences) are supporting a correct claim, we expect them to be credible. Both of these nodes are connected to source 1, where source 1 is connected to domain 0. There are two factors that affect the credibility score of source 1. First, we expect the credibility score of source 1 to be high since

it is providing two credible evidences. However, on the other hand, source 1 is providing an evidence 5 which supports claim 9. There are two other evidences (6 and 7) which are also connected to claim 9. Unlike evidence 5, evidences 6 and 7 are against claim 9. This contradiction decreases the credibility of evidences 6 and 7. Both of these evidences are connected to domain 0, hence we consecutively decrease the credibility of node 0. This decrease in the credibility score of source 2 also affects other nodes in the graph. The scores that are assigned by ClaimEval in graph (a), follow our inference.

2. **Graph (b)**: The main reason that the credibility scores of nodes 1, 3, 4, and 5 have been decreased in graph (a), is because all of the nodes in the graph are connected to the domain node 0, where domain 0 is providing contradictory information about different claims. The fact that source 2 is not credible is also propagated to node 0, which is accordingly affecting the credibility of source 1 and all the evidences extracted from source 1. Graph (b) shows the scores when we connect the evidences $\{4, 5, 6\}$ and $\{7, 8\}$ to different domain nodes 0 and 1. Since they are connected to different domains, the contradictory information that comes from evidences 7 and 8 are only affecting nodes 3 and 1. As the result, ClaimEval is assigning the full credibility score to all the nodes connected to node 0, and credibility scores of nodes 1, 3, 7, and 8 are all set to zero.

3. **Graph (c)**: The credibility score of source 3 and domain 1 are significantly low in graph (b), since all the evidences that they are providing are in contradiction with the information that we believe is *true* (training data). Now assume that we change the value of evidence 8 from *con* to *pro*. In this case, one of the evidences provided by source 3 is supporting claim 10, and hence more consistent with the value of evidence 6. This will increase the credibility of evidences 7 and 8, source 3, and domain 1. However, the fact that source 3 (which is now somewhat credible) is also providing a *con* argument for claim 10, decreases the credibility of node 6 and all the evidences connected to source 2.

4. **Graph (d)**: Since in graph (c), source 3 was providing both a *con* and *pro* argument, the credibility of other sources connected to the claim 10 are decreased. If we

slightly change this graph by connecting evidences 7 and 8 to two different sources (graph (d)), we will observe that the contradictory information provided by evidence 9 is only affecting the sources that it is connected to.

5. **Graph (e)**: One of the main advantages of ClaimEval is to give flexibility to users to assign the *true* label to any node in the graph. Graph (e) shows an example when both a domain and a claim are annotated as training data. In this graph, both nodes 1 and 11 are annotated as credible. Annotating both these nodes to be credible leads to an inconsistency. On one side, source 3 which has gained credibility by correctly assessing the truth of node 11, is supporting claim 12, and on the other side, source 4 which is directly connected to credible domain 1, is against claim 12. It can be seen that ClaimEval has correctly assigned the truth value of 0.5 to node 12.

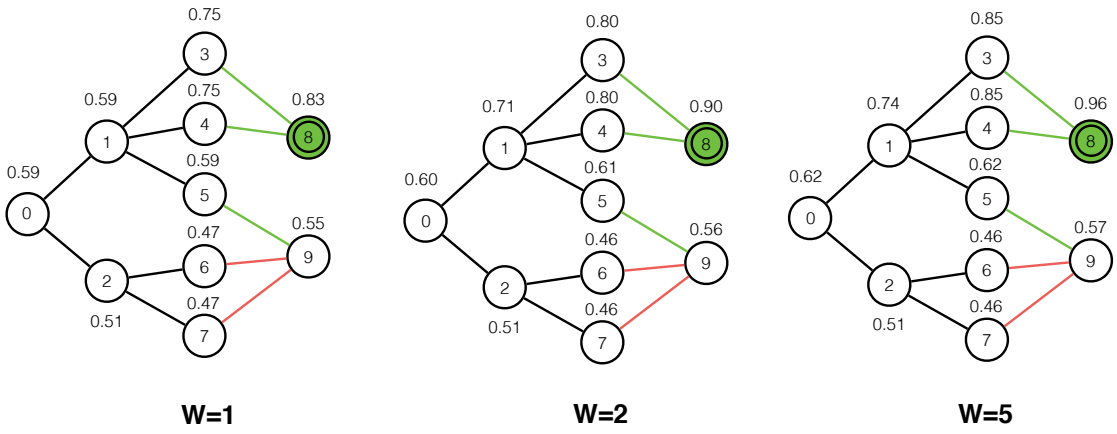


Figure 4.3: Scores assigned by ClaimEval where we use different values for the weight of training data (denoted by W).

Figure 4.3 shows the result of ClaimEval when different weights are used for Equations 4.4 and 4.5 (training data). The graphs show that as we increase the weight of the training data, the assigned scores will be more biased toward the labels of training data.

4.3 Experimental Evaluation

The main hypothesis behind our research is that by using the structure of the CA graph and considering the information provided by different sources, we are able to (i) measure the trustworthiness of the different sources of information (i.e., websites), and (ii) assert the truthfulness of a set of given claims. One way to measure the effectiveness of our approach in achieving these goals, is to show that incorporating the trust score will improve the accuracy of ClaimEval in predicting the truth of claims. In order to validate this, we compare the accuracy of claims made by ClaimEval to different baseline approaches where they may or may not take the credibility score of sources into the account.

We design our experiments to study three main questions. First, what is the accuracy of ClaimEval in predicting the truth of a variety of different claims? Second, what is the accuracy of ClaimEval compared to different baseline approaches? Third, how is the accuracy of ClaimEval affected for different setting when the structure of the CA graph is different?

4.3.1 Setup

ClaimEval is tested on nine different sets of categories (predicates with one argument), althought our approach is general and can be used for predicates with any number of arguments. Table 4.4 lists all the categories in the leftmost column. For each category, about 5-15 seed examples are used to train a SVM-based evidence classifier (used to annotate evidences with *pro* or *con* labels), and about 15-40 examples are used as test data. To train the evidence classifier, we use the top ten pages returned by Bing, and all the evidences extracted from the returned search pages. 0.5 is used as the confidence threshold for all experiments. The methodology used to obtain the train and test data for each category are as following:

- **High Ranked University**: To determine if a given university is a high-ranked university. The ground truth is obtained from Webometrics Ranking of World Univer-

sities [1] website, by selecting 55 universities from the top and 55 universities from the bottom of the ranked list of 11,800 universities. 15 positive and 15 negative instances are used to train the classifier, 40 positive and 40 negative instances are used as the test data.

- **Company with Stock Growth**: To determine if a stock price of a company is increasing. The ground truth is obtained from the Barchart website [2], which is a leading provider of intraday stock and commodities. The list of companies is chosen by selecting a set of companies whose stock prices have increased at least by 100% over the past 10 years, and a set of companies whose stock prices have decreased by 100% over the past 10 years. 15 positive and 15 negative instances are used to train the classifier, 40 positive and 40 negative instances are used as the test data.

- **Top Conference In Computer Science**: To determine if a given CS conference is considered as a top conference. The ground truth is obtained from Microsoft Academic Search Ranking [3], by selecting the top and bottom ranked conferences in Computer Science (CS) (among 3,523). 15 positive and 15 negative instances are used to train the classifier, 40 positive and 40 negative instances are used as the test data.

- **Top Journal in Computer Science**: To determine if a given CS journal is considered a top journal. The data for this category is obtained by choosing the top and bottom CS journals from Microsoft Academic Search Ranking. 15 positive and 15 negative instances are used to train the classifier, as the test data, we use 40 positive and 40 negative instances.

- **Healthy Food**: To determine if a given food is healthy or not. The list of foods are selected from trustworthy websites such as the U.S. Food and Drug Administration [4] and WebMD. 15 positive and 15 negative instances are used to train the classifier, 40 positive and 40 negative instances are used as the test data.

[1] http://www.webometrics.info/
[2] http://www.barchart.com/
[3] http://academic.research.microsoft.com/
[4] http://www.fda.gov/

- **High GDP Growth**: To determine if a given country had a high Gross Domestic Product (GDP) growth. GDP is usually one of the important factors used to determine how good a country's economy is. The positive instances are chosen from the countries with the GDP of higher than 5% between 2006-2010, and the negative instances are chosen from the countries that have had negative GDP between 2006-2010. We have used 5 positive and 5 negative training instances, and 10 positive and 10 negative test instances. The instances are extracted from http://worldbank.org.

- **High HDI Growth**: To determine if a given country has a high Human Development Index (HDI) growth. Since 1990, the HDI has been used as an important measure of progress in human development. HDI is a composite index of life expectancy, years of schooling and income. The report published by United Nations Development Programme (UNDP) [5], has categorized countries to different groups based on their HDI score. The positive instances for this predicate are chosen from the countries that are associated with the "very high human development" group, and the negative instances are chosen from the "low human development" group. In our experiments, we have used 10 positive and 10 negative training instances, and 20 positive and 20 negative test instances. All the instances are chosen from the UNDP official report http://hdr.undp.org/sites/default/files/hdr14-report-en-1.pdf.

- **High Crime Rate Cities:** To determine if a given U.S. city has a high crime (offenses known to law enforcement) rate. The test and training instances are chosen from data of all the U.S. cities reported in the FBI report in 2012: http://www.fbi.gov/about-us/cjis/ucr/crime-in-the-u.s/2012/crime-in-the-u.s.-2012/tables/8tabledatadecpdf/table˙8˙offenses˙known˙to˙law˙enforcement˙by˙state˙by˙city˙2012.xls The positive instances are chosen from the cities with the highest crime rate (from the top of the list, where the crime rate is higher than 1.5%) with more than 10K population. The negative instances are chosen from the list of lowest crime rate cities which have a population of more than 10K and a crime rate of less than 0.015%. 15 positive and 15 negative instances are used as training data. 30 positive and 30 negative instances are used as test data.

[5] http://hdr.undp.org/sites/default/files/hdr14-report-en-1.pdf

- **Top Soccer Club Teams:** To determine if a European club team has a high rank. The positive and negative instances are chosen from the list of teams ranked by Union of European Football Associations (UEFA). 12 positive (highest ranked club teams) and 12 negative (lowest ranked club teams) instances are chosen as training data, and 50 instances are chosen as the test data (25 positive and 25 negative).

4.3.2 Baseline Approaches

We compare the accuracy of ClaimEval to the following Fact-Finding approaches: Majority Vote (MV), Generalized Sums (GS) which is based on Hubs and Authorities algorithm [Kleinberg, 1999], Truth Finder (TF) [Yin et al., 2007], Average-Log (AL) [Pasternack and Roth, 2011], Generalized Investment (GI) [Pasternack and Roth, 2011], and Pooled-Investment (PI) [Pasternack and Roth, 2010]. All of these baseline approaches operate over a bipartite graph construction (two layers: sources and claims) instead of CA graph. In order to compare ClaimEval with these approaches, we construct the equivalent bipartite graph from a CA graph by marginalizing over evidence nodes and dropping the domain nodes. Apart from MV, all other baselines make use of the available labeled claims.

The fact-finding algorithms used as the baseline of our comparison can be summarized as following. Assume that we have set of sources, S, and a set of claims, C. Each claim $c \in C$ is asserted by a set of sources S_c, and each source $s \in S$ is asserted a set of claims C_s. The sources and claims can be viewed as a bipartite graph where an edge exists between c and s, if $s \in C_s$ and $c \in C_s$. At each iteration i of a fact-finding algorithm, the credibility of a source s, denoted by $T^i(s)$ is estimated by our current beliefs about the truth of claims, denoted by B^{i-1}. Similarly, the correctness of claim c (i.e., belief in claim c, denoted by $B^i(c)$) is estimated using the trustworthiness of sources that assert claim c (denoted by T^{i-1}). The values of B and T are iteratively updated until convergence or a stop condition. Initially, the initial values of beliefs B^0 are set using our prior knowledge (i.e., equal to 0.5, if no prior knowledge is available).

In order to compare ClaimEval with iterative fact-finding algorithms, we first need to convert the CA graph to a bipartite graph with two layers: *sources* and *claims*. To convert

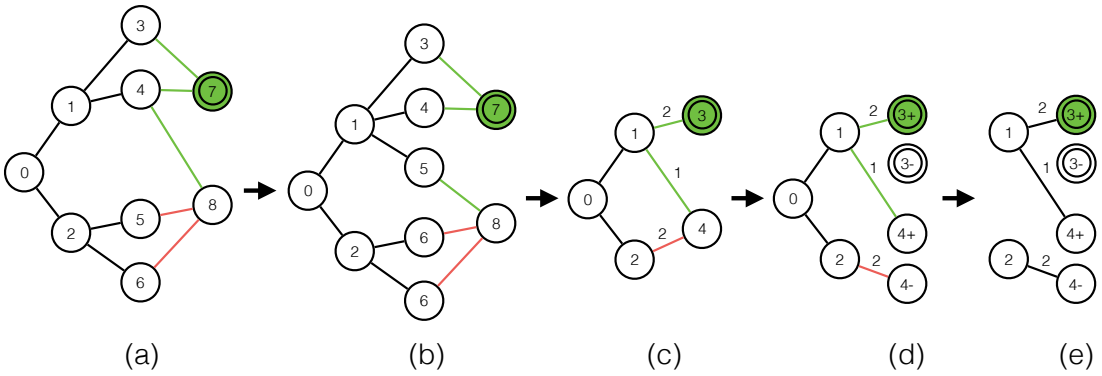


Figure 4.4: Step by step demonstration of how a Credibility Assessment graph is converted to a bipartite graph, which is then used as an input of baseline fact-finding algorithms.

the CA graph, we follow these steps (each step is demonstrated in Figure 4.5):

1. For each evidence e that is connected to $K > 1$ number of claims, replace it with K evidences, identical to e, and connect each of the evidences to only one of the K claims. At the end of this step, the degree of each evidence node is equal to two (Figure 4.5(b)).

2. For every triple of $\langle s, e, c \rangle$, if s is connected to e, and e is connected to c, remove e and directly connect s to c. The edge from s to c has the same label as the edge from e to c. The weight of the edge from s to c is equal to the total number of edges that exist between s and c (Figure 4.5(c)).

3. Replace each claim node c, with two nodes c^+ and c^-. Connect all the *pro* evidences of c to the node c^+, and all the *con* evidences of c to the node c^- (Figure 4.5(d)).

4. Remove the domain layer and all the edges connected to it (Figure 4.5(e)).

The input of a fact-finding algorithm is a bipartite graph (e.g., Figure 4.5(e)) and the output is a set of scores assigned to each claim in the graph (score values can be larger than 0). The truth value of each claim c is determined by comparing the belief values $B^i(c^+)$ and $B^i(c^-)$, if $B^i(c^+) > B^i(c^-)$ then the claim is true, and otherwise is false. A

fact-finding algorithm is specified by three main components: (i) the update function for T, (ii) the update function for B, and (iii) the prior values for B^0. For all of the fact-finding algorithms, the prior value (B^0) is set to 0.5. The baseline fact-finding algorithms are:

1. **Majority Vote (MV)**: The truth value of a claim c is calculated by the number of sources that asserts c to be true, divided by the total number of sources asserting the value of c as either true or false. More formally,

$$B(c) = \frac{\sum_{s \in S_c} w(s,c)}{\sum_{s \in S_{c^-} \cup S_{c^+}} w(s,c)} \tag{4.6}$$

   where $w(s,c)$ is the weight of edge that connects node s to c.

2. **Generalized Sums (GS)**: The Sums fact-finder is based on the Hubs and Authorities algorithm [Kleinberg, 1999; Pasternack and Roth, 2011] and is defined as follows:

$$T^i(s) = \sum_{c \in C_s} \omega(s,c) B^{i-1}(c) \quad B^i(c) = \sum_{s \in S_c} \omega(s,c) T^i(s) \tag{4.7}$$

3. **Average-Log (AL)**: AverageLog [Pasternack and Roth, 2011] algorithms use the same B function as the GS algorithm. The T update function is defined as follows:

$$T^i(s) = \log\left(\sum_{c \in C_s} \omega(s,c)\right) \cdot \frac{\sum_{c \in C_s} \omega(s,c) B^{i-1}(c)}{\sum_{c \in C_s} \omega(s,c)} \tag{4.8}$$

4. **TruthFinder (TF)**: The update functions for the TruthFinder [Yin et al., 2007] is defined as follows:

$$T^i(s) = \frac{\sum_{c \in C_s} B^{i-1}(c)}{|C_s|}$$
$$B^i(c) = 1 - \prod_{s \in S_c} \left(1 - T^i(s)\right)$$

(4.9)

5. **Generalized Investment (GI)**: The update functions for the Generalized Investment algorithm [Pasternack and Roth, 2011] is defined as follows, where $\mathcal{G} = x^{1.2}$:

$$T^i(s) = \sum_{c \in C_s} \frac{\omega(s,c) B^{i-1}(c) T^{i-1}(s)}{\sum_{c \in C_s} \omega(s,c) \cdot \sum_{r \in S_c} \frac{\omega(r,c) T^{i-1}(r)}{\sum_{b \in C_r} \omega(r,b)}}$$
$$B^i(c) = \mathcal{G}\left(\sum_{s \in S_c} \frac{\omega(s,c) T^i(s)}{\sum_{c \in C_s} \omega(s,c)}\right)$$

(4.10)

6. **Pooled Investment (PI)**: The update functions for the Pooled Investment algorithm [Pasternack and Roth, 2011] is similar to the GI algorithm. The T function remains the same Given $H^i(c) = \sum_{s \in S_c} \frac{T^i(s)}{|C_s|}$, the B function is defined as follows:

$$B^i(c) = H^i(c) \cdot \frac{\mathcal{G}(H^i(c))}{\sum_{d \in M_c} \mathcal{G}(H^i(d))}$$

(4.11)

4.3.3 Comparison to Baselines: Example

As described in the introduction, one of the main disadvantages of current fact-finding algorithms is their bias toward increasing the credibility of sources that assert many claims.

Figure 4.5 shows an example of a CA graph where a source (node 2 in the figure) is asserting a claim by providing many evidences. Most of the fact-finding algorithms provide a biased result toward classifying the value of claim 13 as *false* (since it is asserted by many *con* evidences). In contrast to the current fact-finding algorithms, ClaimEval is able to infer that node 2 has a low credibility score since it is providing a contradictory information regarding claim 12 (which we know its truth value), and to automatically propagate the low credibility score of source 2 in order to reduce its affect on claim 13.

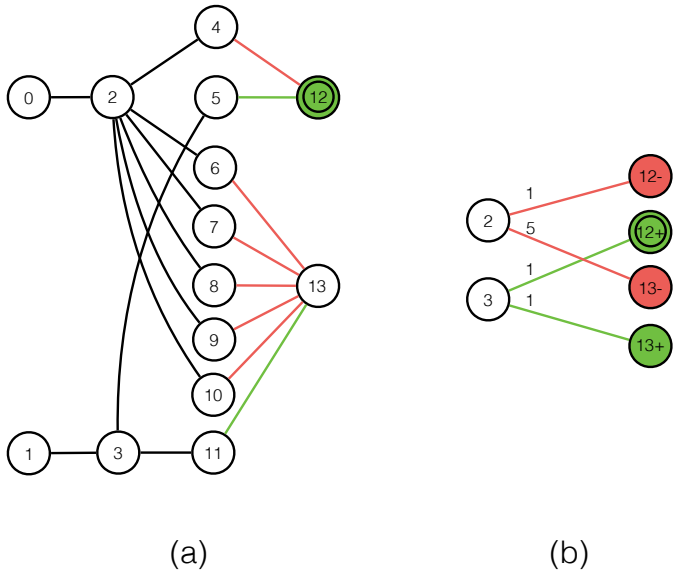


Figure 4.5: (a) An example of a Credibility Assessment graph, (b) and its equivalent bipartite graph.

Figure 4.6 shows an example of a CA graph (and its equivalent bipartite graphs), where a domain node 0 (through webpage node 2) is asserting overwhelmingly many evidences against claim node 13 (negative evidence is marked by red line in the figure). Claim node 12 with concentric circles is known to be correct a-priori. Correctness of the other claim node 13 needs to be evaluated. Based on these input information, judgments of a human annotator is shown on top of each node (the same as the scores assigned by ClaimEval). Most of the fact-finding algorithms (including Majority Vote, AverageLog, and Generalized Sums) are biased towards favoring domain nodes with many evidences. In this example, these algorithms overestimate the credibility of nodes 0 and 2, and incorrectly classify

claim 13 as incorrect (as all evidences from source node 2 oppose claim 13). In contrast, ClaimEval is able to infer that nodes 0 and 2 have low credibility since they provide contradictory information regarding claim 12. ClaimEval then uses this reduced credibility to correctly classify claim 13 as true. This examples provides qualitative evidence of how ClaimEval is able to overcome bias of many existing fact-finding algorithms. Among all the baseline approaches, only the Generalized Investment approach has been able to correctly classify claim 13 as *true*. However it has still assigned a large value to node 13^- (compared to the value of node 13^+). The value of node 13^- increases as node 2 asserts more number of claims.

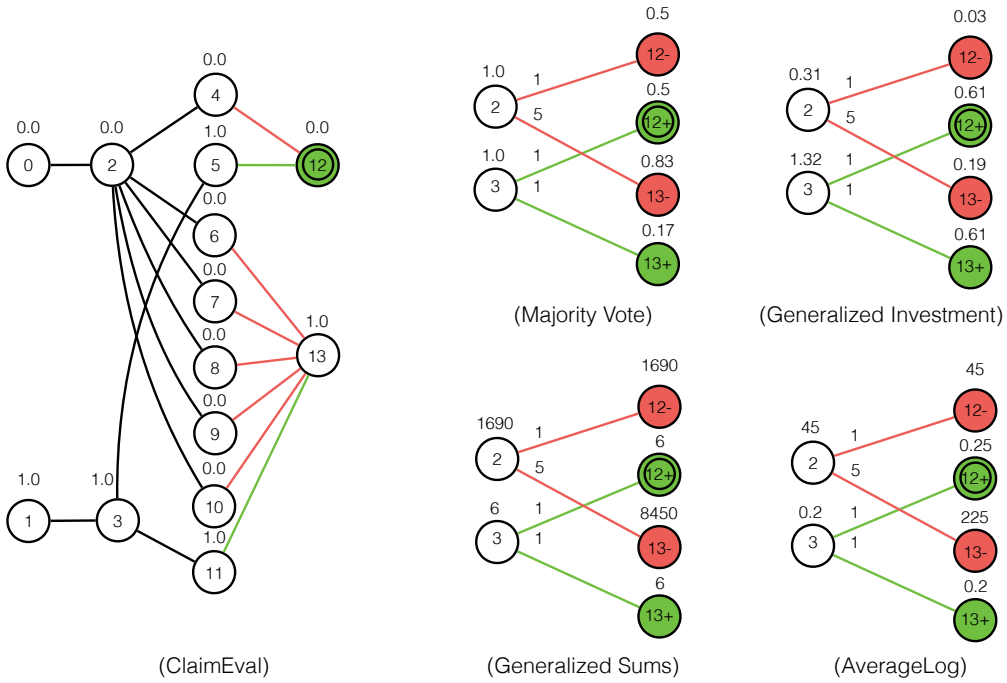


Figure 4.6: An example CA graph, with the scores assigned by different approaches marked on top of each node. While baselines such as Majority Vote, Generalized Sums and AverageLog overfit by over-trusting domains with many evidences (e.g., node 13), ClaimEval is able to find the annotator judgments.

4.3.4 Comparison to Baselines: Experimental Results

Tables 4.4 shows the accuracy of the Majority Vote (MV), Generalized Sums (GS), Average-Log (AL), Generalized Investment (GI), and ClaimEval approaches, for five different categories. The experiments are obtained when ClaimEval uses the SVM classifier, and 60 evidences are randomly extracted for each claim from the first 10 pages returned by Bing [6]. The first column of the table shows the categories that are used in the experiments, and the next columns show the accuracy of each baseline approach.

Category	**MV**	**GS**	**TF**	**AL**	**GI**	**PI**	**CE**
Healthy Food	0.89	0.89	0.69	0.86	0.89	0.89	**0.91**
Company with Stock Growth	0.62	0.65	0.65	0.62	0.62	0.60	**0.72**
High Ranked Universities	0.80	0.82	0.73	**0.85**	0.80	0.80	**0.85**
Top CS Journals	0.74	0.71	**0.82**	0.67	0.71	0.71	0.79
Top CS Conferences	0.53	0.55	0.58	0.57	0.53	0.53	**0.68**
High GDP Growth	0.60	**0.70**	0.50	**0.70**	0.6	0.50	0.60
High HDI Growth	0.81	0.53	0.65	0.63	0.81	**0.86**	0.76
High Crime Rate Cities	0.67	0.63	0.67	0.60	**0.80**	**0.80**	**0.80**
Top Soccer Club Teams	0.65	0.62	0.65	0.65	**0.69**	0.62	**0.69**
Average	0.71	0.68	0.66	0.69	0.72	0.71	**0.76**

Table 4.4: The accuracy of the Majority Vote (MV), Generalized Sums (GS), TruthFinder (TF), Average-Log (AL), Generalized Investment (GI), Pooled-Investment (PI), and ClaimEval (CE) techniques in predicting the truth values for different categories of claims. The maximum value of each row is shown in bold. ClaimEval, the proposed system, achieves the best overall performance.

Table 4.4 shows that the average accuracy of ClaimEval is higher than all other baseline approaches. We observe that the performance of fact-finding algorithms may be category-specific, which is consistent with similar observations in [Pasternack and Roth, 2011]. Overall, ClaimEval, the proposed system, achieves best performance in 6 out of the 9 categories, and on average out-performs all other state-of-the-art baselines.

[6] www.bing.com

Change in Performance with Increasing Evidence Size: Figure 4.7 shows accuracy of different methods when increasing the number of evidences that are used. As we increase the number of evidences, ClaimEval consistently outperforms all other methods. Among the baselines, *Generalized Investment* is more successful in making use of the increased evidence size, which is consistent with prior research [Pasternack and Roth, 2011]. *TruthFinder*'s performance peaks with 30 evidences, but its performance degrades with more evidences due to over-fitting.

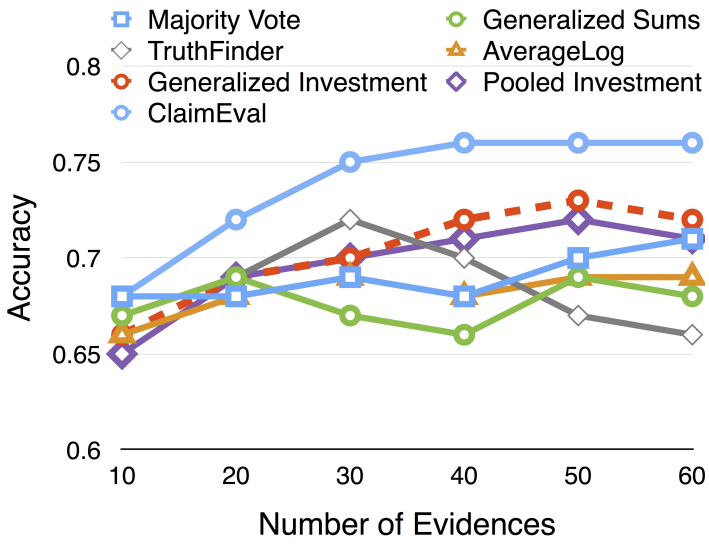


Figure 4.7: Performance of different systems when increasing amounts of evidence are available. ClaimEval, the proposed system (top-most plot), is best able to exploit additional evidence achieving the best overall performance.

4.4 Summary

This chapter introduced ClaimEval, a novel approach for credibility assignment, where the prior knowledge of credibility inference can be provided in a flexible and principled approach. In particular, we described in detail how to build the Credibility Assessment Graph (CAG), represent the credibility propagation rules, and use Probabilistic Soft Logic (PSL) to represent prior knowledge and find credibility and truthfulness of a set of sources

and evidences. We experimentally showed the effectiveness of the ClaimEval approach by outperforming the state-of-the-art fact-finding approaches: majority vote, generalized sums, truth-finder, average-log, and generalized investment approaches. Comparison of the average accuracy of ClaimEval and baseline approaches shows that ClaimEval greatly outperforms all other baselines.

In contrast to previous approaches for claim evaluation, given a set of claims, ClaimEval identifies a set of relevant sources and evidences within them which might support or refute the claims, estimates the credibility of those sources, and uses those credibility estimates to evaluate the correctness of the claims – all in a single integrated system. Unlike other approaches, in ClaimEval, it is easy to state the prior knowledge, the convergence of the credibility assignment algorithm is guaranteed, and the assigned credibility scores are interpretable.

Chapter 5

Using the Web to Interactively Learn to Find Objects

In the previous chapters, we explained the detail of our information extraction systems. One of the main motivations behind these approaches was to address information extraction tasks that are initiated as queries from either automated agents or humans. In this chapter, we show how our information extraction systems, in particular OpenEval, can be used to provide knowledge to a real mobile robot (CoBot).

We begin in Section 5.1 with motivating the problem of actively querying the Web to learn new background knowledge about the physical environment, in particular to learn the knowledge that is necessary for finding and delivering an arbitrary object in an environment. We explain the details of our approach, ObjectEval, in Section 5.2. Section 5.3 presents the detail of our experiments to evaluate ObjectEval system. Finally, Section 5.4 presents the summary of this chapter.

5.1 Introduction

Our aim in this chapter is to make mobile robots that are able to intelligently perform tasks by combining the retrieved knowledge by OpenEval and use it in a mobile robot. In this chapter, we investigate a *find and deliver* task, where a person specifies an object in open-ended natural language (e.g., "coffee") and the robot will find and deliver the object to a specified destination (e.g., "GHC-8125"). This is a challenging problem because robots have limited perceptual abilities and people use highly variable language when specifying a task. For example, since our robot only has access to the type of a room (e.g., "office" or "kitchen"), it will need to learn the relationship between a query object (e.g., "coffee") and these room types. In addition, since a person may ask the robot to find any of thousands of objects, including a "coffee," "pen," or "papers," the robot will need to learn these relationships over all possible query objects. The find and deliver task is easy for humans, who have learned a large set of background knowledge from experience with the environment; for robots, which have limited access to such knowledge, it is much more challenging.

We introduce an approach, called ObjectEval, which addresses the challenges of the find and deliver task by querying the Web using the OpenEval approach. In this chapter, we train OpenEval on a single predicate that describes when an object can be found in a location: *locationHasObject(X, Y)*. The training data takes the form of a small number of these predicate instances, which define example object/location pairs. OpenEval uses each of these training instances to create a search query which is then sent to a search engine. The results of this query are a set of context-based snippets, which OpenEval uses to train a model of the types of objects that can be found in each location. Some examples of documents returned by search queries can be seen in Figure 5.1.

The probability that is computed by OpenEval is then dynamically incorporated into a utility function, which takes into account the travel distance to a location, the number of human interactions required to get to a location, and the observation of the object during previous executions at that location. ObjectEval then infers the maximum-utility plan corresponding to a sequence of locations it should visit, asks a human to provide it with the object, and then takes the object to a destination.

Instances of predicates	Probability	Search Results
locationHasObject (bathroom,hair dryer)	$p = 0.89$	**Hair Dryer** Caddy: Clear the clutter off your **bathroom** counter and still keep everything you need. Get your **hair dryer** off the countertop and add a decorative note to your **bathroom** at the same time.
locationHasObject (kitchen,food)	$p = 0.92$	Everything you need to make your **kitchen** a functional and fashionable place to cook is here at the **food** section Momma's Soul **Food Kitchen** serves delicious soul food lunches and dinners. Located at 3319C Raeford Road, Fayetteville, NC.
locationHasObject (printer room,fax machine)	$p = 0.81$	There is a **fax machine** in the **printer room** at work that I use about once a year to Keep the **printer room** neat! KEEP THE PRINTER ROOM NEAT! Stop using the **fax machine** for personal business!
locationHasObject (kitchen,coffee)	$p = 0.85$	Espresso Machines; **Coffee** Makers French Presses from **Coffee** and **Kitchen**. Create your own **kitchen** bistro with our **Coffee** House kitchen collection!

Figure 5.1: Examples of snippets returned by search queries for the predicate locationHasObject. On the left are some instances of the predicate objectInLocation that we have queried the system with. In the middle, is the associated probability of this predicate instance according to OpenEval. On the right, are two documents returned by a web search that are used by OpenEval to evaluate the probability of the predicate.

We evaluate ObjectEval in three ways. First, we show that ObjectEval is able to predict the location of novel objects against a baseline that is similar to Kollar and Roy [Kollar and Roy, 2009]. Second, we show that ObjectEval is able to efficiently and automatically find novel objects in a realistic simulated environment consisting of 290 spaces of an office building given only a topological map and the space types (e.g., "office," "bathroom" etc.). We further show that ObjectEval can improve its performance by learning from feedback it gets about the location of objects. Third, we demonstrate a find and fetch task on CoBot (Figure 5.2), a mobile office assistant, showing that our system can enable CoBot to find objects in a real-world environment. Our system can successfully find a set

of objects, including the “papers,” “coffee,” “toner,” and “laptop.”

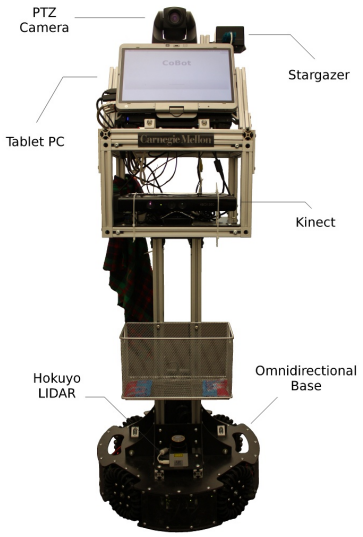
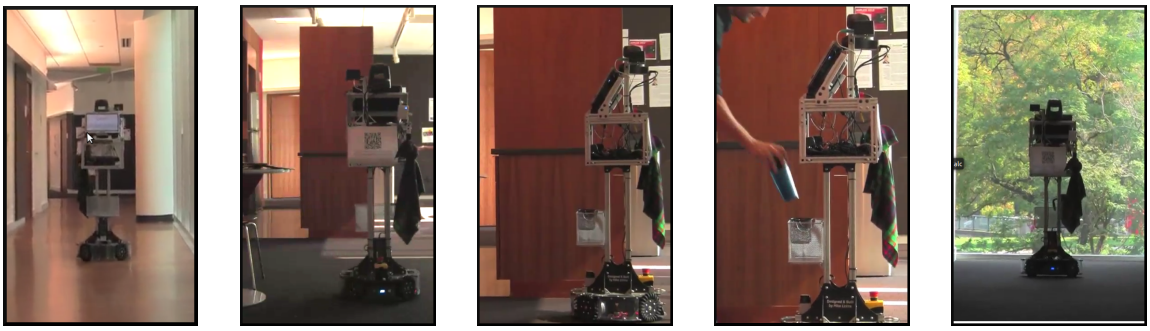


Figure 5.2: A robotic platform that we have used to find and fetch objects.

5.2 ObjectEval

Our approach, called ObjectEval, enables a robot with limited sensing to search for an object. By using symbiotic autonomy, the robot is able to ask people to help it perform tasks, including manipulation and object detection [Rosenthal et al., 2010; Biswas and Veloso, 2012]. To find an object, the robot must therefore (1) receive a command to find an object (e.g., “coffee”) and take it to a destination (e.g., room 7001), (2) compute a sequence of locations to visit by maximizing long-term utility, (3) visit a location, (4) ask a person to retrieve the object and finally (5) if there, deliver the object to the destination or if not, go to the next location to look for the object. An example of our robot finding a “coffee” can be seen in Figure 5.3.



(a) Receive command (b) Go to location (c) Ask for "coffee" (d) Get "coffee" (e) Deliver object

Figure 5.3: An example of our robot searching for an object. In (a) the system gets a query to find a "coffee" and take it to room 7001. In (b) it goes to the nearest kitchen. In (c), it asks a person to place a coffee on it. In (d), it gets the coffee and the person says that the robot has the object. In (e), the robot delivers the object to its destination.

5.2.1 Model

ObjectEval takes as input an object name (e.g., papers) and a destination room (e.g., room 8120), and returns a plan consisting of locations that the robot should visit. Finding objects requires trading off different objectives including: the number of interactions with people, the distance traveled, the existence of objects at previously visited locations, and the probability of finding an object in a location. ObjectEval combines these objectives into a utility function that, when maximized, generates a plan that the robot can execute to find an object effectively. If O is an object name (e.g., "papers"), and U is the utility function, then the problem can be formulated as finding the plan that maximizes the utility:

$$\arg\max_{\text{plan}} U(\text{plan}|O) \tag{5.1}$$

The plan is broken down into a sequence of steps (plan_i), each of which visits a location and asks for an object from a person. The robot receives a reward (R) when it executes ith step of the plan. The current step in the plan is successful with probability $p(\text{plan}_i|O)$.

$$U(\text{plan}|O) = \sum_{i=1}^{N} p(\text{plan}_i|O) \times R(\text{plan}_i, O) \tag{5.2}$$

In order to capture the objective of finding objects quickly, the reward at each step is broken down into three components:

$$R(\text{plan}_i, O) = D(\text{plan}_i) \times I(\text{plan}_i) \times F(\text{plan}_i, O) \tag{5.3}$$

Since the robot should consider plans that travel as little as possible, we include the reward D, which is dependent on the distance the robot travels. D is computed by subtracting the distance traveled from the maximum distance the robot could travel. Since people are used as a part of the search process to find and manipulate objects, we include the reward I, which is dependent on the number of interactions that the robot has with a person. I is computed by subtracting the number of interactions required to search a location for an object from the maximum number of interactions the robot will need to search for any location. Finally, in order to take advantage of feedback from people, we include the reward F, which uses previous searches to help search for objects. The value of F is 1 if a query object has been seen at the search location, 0.5 if the location has not been explored, and 0 if it is known not to exist there. Although F is fixed in this paper, learning a dynamic model for how objects move would enable ObjectEval to handle cases where the query object moves between different locations in the environment.

The second component of Equation 5.2 requires us to compute the probability of a part of the plan. As a proxy for the probability of the plan, we use the probability that the location at the ith step of the plan will contain an object given that the object was not seen at the previously visited locations in the plan. If l_j is multinomial over location types (e.g., "office," "printer room," "bathroom") and O is the query object, then the we can compute this probability as:

$$p(\text{plan}_i|O) \approx \left[\prod_{j=1}^{i-1} (1 - p(l_j|O))\right] \times p(l_i|O) \tag{5.4}$$

In order to find the plan with a maximum utility, the robot must be able to compute $p(l_i|O)$. This term connects a query object O (e.g., "papers") to a location type in the environment (e.g., "printer room"). Connecting a query word for an object to a place where the robot can find the object is challenging because there are thousands of different object names people might use. We calculate the probability $p(l_i|O)$ by querying the Web

using OpenEval approach for the validity of the predicate *locationHasObject(l,O)* over all location types l. For example:

$$p(l_j = \text{kitchen} | O = \text{coffee}) \triangleq p(\text{locationHasObject(kitchen, coffee)}) \quad (5.5)$$

In Chapter 2 we explained the details of the OpenEval approach. In the next section, we provide a brief overview of how we can use the OpenEval approach to obtain the probability of instances of the predicate *locationHasObject*.

5.2.2 Querying the Web using OpenEval

The World Wide Web (WWW) contains an enormous amount of semantic information that might be useful for robots. In this paper, we investigate the use of the semantic information on the Web to predict the location of objects in real-world environments. We expect that objects physically present in a location will be found frequently on the Web. For example, one of the top search results for the object "paper" and the location "printer room" is, "There is no more **paper** in the **printer room**, where can I find some more?" For objects unrelated to the location, such as "papers" and "elevator" there are fewer pages which often describe less sensical events such as, "Call for **Papers**, The International Space **Elevator** Consortium (ISEC) invites you to join us in Washington State." Therefore, we expect that the word patterns for related terms will be predictive, while un-related terms will be less predictive. Figure 5.4 shows example of text snippets that are found on the Web for object "papers" and locations "printer room" and "bathroom."

ObjectEval will compute the probability from Equation 5.5 by converting input commands to predicate instances in first-order logic, such as *locationHasObject(papers, printer room)*. It then uses OpenEval in order to search the Web query and retrieve information from hundreds or thousands of the most relevant webpages that relate these terms. The text on the webpages that is most relevant to a predicate instance is then extracted by OpenEval. Figure 5.4 shows an example of the text snippet that is found in one of the returned webpages for query {*"Papers" "Printer Room"*}. Some of the extracted features include: {*documents*, *printed, office, unfetched*}.

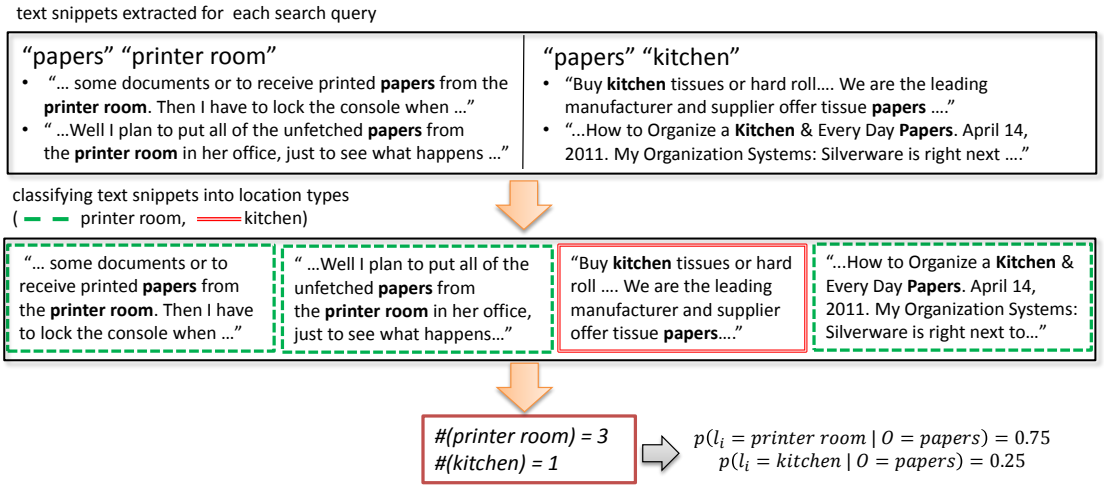


Figure 5.4: An example of how ObjectEval queries the Web to find the maximum probability location of "papers." In this example, two location types exist: printer room and kitchen. A Web search for "papers," "printer room" and "papers," "kitchen" returns a set of webpages from which text snippets are extracted. The features of these text snippets are then categorized as one of the location types. The red border indicates that the text snippet was categorized as a "kitchen" and the green border indicates it was categorized as a "printer room." The frequency of the resulting categorization is then used to compute the maximum likelihood probability of each location given a query object.

Thus, given a query object O, ObjectEval will create a set of predicate instances over all locations l in the environment (Figure 5.4). If *papers* is given as an object query and we have only two location types in the environment, *printer room* and *kitchen*, then using OpenEval, it formulates two search queries: {"papers" "printer room"} and {"papers" "kitchen"}. Each search query then returns a set of the highest ranked webpages from a Web search engine[1]. Text snippets are then extracted for each of the webpages as shown in the top box of Figure 5.4. The result of classification on all of the text snippets is shown by two colors: green means that the text snippet is classified to location "kitchen" and red means that it is classified as "printer room". In total, three of the text snippets are classified as "printer room" and one is classified as "kitchen". This leads to a probability

[1]such as Bing, www.bing.com

of 0.75 (3/4) for object "papers" to be found in "printer room" and a probability of 0.25 (1/4) for "papers" to be found in the "kitchen."

5.2.3 Inferring a Plan

ObjectEval searches over candidate plans to maximize the utility function in Equation 5.1. We use a beam search with a beam width of 10 and search up to depth $N = 10$ in the search tree. At each step of the search, a new plan step $plan_i$ is added to the overall plan. Each plan step $plan_i$ can visit any of the locations in the test environment to find an object. The beam search disallows loops by preventing plans from revisiting previously visited locations.

5.3 Evaluation

ObjectEval is evaluated in four primary ways. First, we query OpenEval to predict the probability distribution $p(l_i|O)$ described in our model over a set of test objects. Second, a set of simulated commands are executed for three floors of an office building. Third, we have demonstrated ObjectEval on our robotic platform. Finally, we explain the detail of integrating ObjectEval into KnoWDiaL system [Perera et al., 2015], an approach for robot learning of task-relevant environmental knowledge from human-robot Dialog and access to the Web.

5.3.1 Predicting the Location of Objects

We have collected a corpus of 134 unique instances for the predicate *locationHasObject* for the "kitchen," "bathroom," "office," and "printer room" locations. These instances were acquired by asking subjects on Amazon's mechanical Turk to look at pictures of each location type and describe objects that tend to reside there. The data is split by randomly choosing 68% of the data for training and 32% for testing. OpenEval is trained and tested by using the first 20 webpages that are returned by the search engine. Table 5.1

shows the result for a subset of the test objects. OpenEval is able to correctly determine the most likely location for most objects. It incorrectly classifies "whiteout" to be found in "bathroom." OpenEval also chooses "bathroom" as the most likely location for "cup". Although this is correct in some environments (e.g. hotels), we generally expect robot to find "cup" in either a "kitchen" or an "office". The results show that by requesting a more specific query such as "coffee cup," OpenEval will change its classification to the "kitchen."

Object	**Location Types**			
	Bathroom	Printer Room	Kitchen	Office
coffee	0.08	0.02	**0.72**	0.18
marker	0.33	**0.53**	0.08	0.06
pen	0.15	0.27	0.23	**0.35**
toner	0.05	**0.87**	0.02	0.06
scissors	0.26	0.01	**0.61**	0.12
whiteout	**0.66**	0.02	0.24	0.08
laptop	0.1	**0.48**	0.08	0.34
papers	0	0.17	0.13	**0.7**
cup	**0.42**	0.1	0.36	0.12
coffee cup	0	0.01	**0.73**	0.27
speakers	0.34	0.06	0.25	**0.35**

Table 5.1: The probability that ObjectEval assigns to different test objects for each location type. The location type with the maximum probability is shown as bold.

OpenEval was then evaluated using precision, recall, and F1 (which is a combination of precision and recall) over this dataset. The ESP baseline replaces web search with a search over tag documents that contain the search terms [von Ahn and Dabbish, 2004] in order to provide a comparison to [Kollar and Roy, 2009]. Figure 5.3.1 shows that the model trained on ESP performs worse than ObjectEval, which likely happens because few locations are tagged in the ESP dataset.

Finally, the speed at which OpenEval learns was evaluated. Figure 5.3.1 shows the F1-

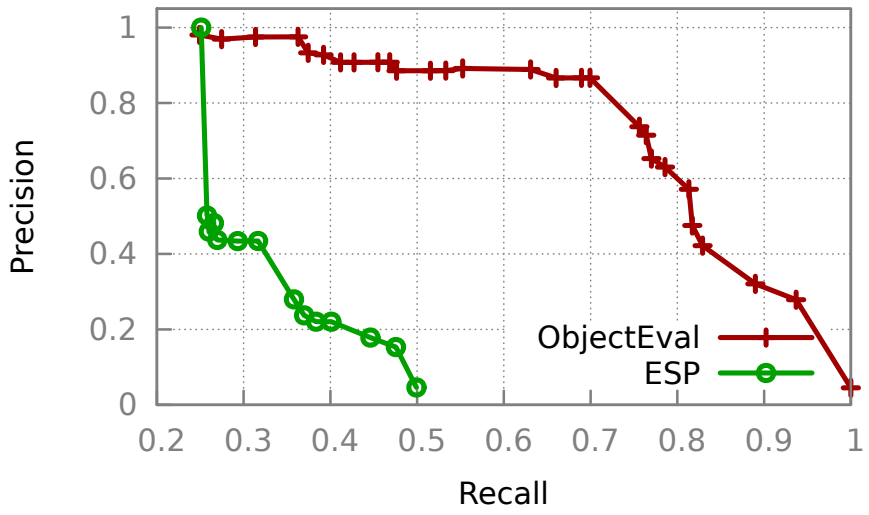


Figure 5.5: The precision/recall curve of OpenEval for the 45 test predicate instances for location types: *bathroom*, *printer room*, *kitchen*, and *office*.

score of OpenEval when increasing the number of predicate instances used for the training. The results are obtained by training on a subset of the training instances and evaluating on all of the test instances. The result, somewhat surprisingly, shows that OpenEval achieves a high F1 value even when it uses a few training examples. For example, it achieves a F1 score of about 60% when it uses only 6 training examples for the training. OpenEval learns quickly because a single training instance could return thousands or millions of webpages. For example, the number of documents referencing "papers" and "printer room" is 61,200 according to Google. This result indicates that OpenEval might be trained even with only a few predicate instances.

5.3.2 Simulated Experiments

We have created a large simulated environment to evaluate how ObjectEval will search for objects. Since the simulator uses exactly the same procedures as the physical robot, the number of interactions (I) will be exactly the same as on the real robot. In general, when the robot asks for an object, a person must answer two questions and when it is moving

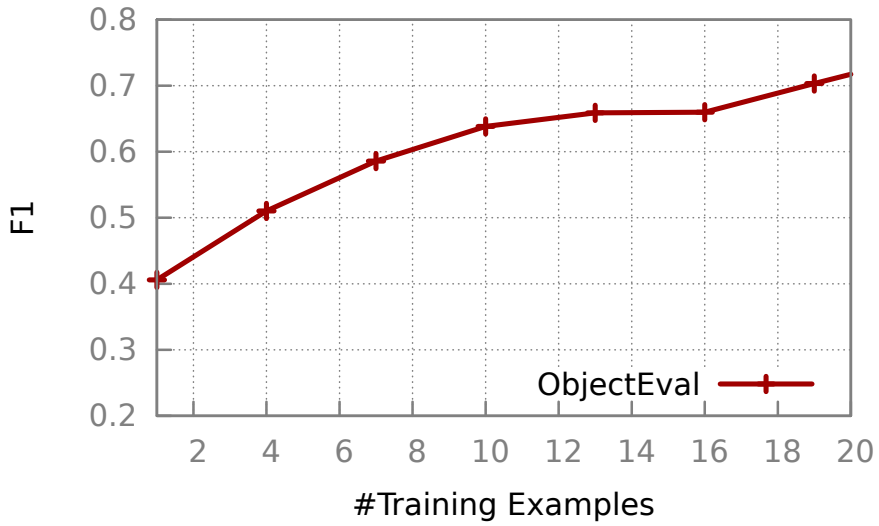


Figure 5.6: The F1-score for the 45 test predicate instances when training on a subset of the training dataset. The figure shows that OpenEval achieves a high F1 score even when it uses a few training examples, and its F1 score significantly increases when more training examples are provided for training.

between floors (using the elevator) a person must answer five questions.

To simulate the objects present in the building, we have created a semantic map of 290 spaces over three floors of an office building that contain names for objects and locations present in each space. This was done by asking subjects on Amazon's Mechanical Turk to label images of 46 rooms with the location and objects present. These labels were transferred to spaces for which we were not able to acquire images by sampling from the data collected from Mechanical Turk. To test the ability of ObjectEval to search for objects, we have selected 80 object types that were not a part of the training set. ObjectEval was given only the location types (e.g., "kitchen" or "printer room") and a map of the environment. For each query object, a random location is chosen as the object delivery destination.

We evaluate ObjectEval in two scenarios: *offline mode* and *interactive mode*. In the offline mode, ObjectEval learns the probability from Equation 5.4 by using a small dataset

of predicate instances consisting of objects and a place where that object can generally be found. In interactive mode, the robot starts performing the find and deliver task in an unknown environment without this training data. By interacting with people, ObjectEval acquires examples of objects and the corresponding place where the object was found. This is then used to learn a model of $p(l_i|O)$ in Equation 5.4. When the robot finds an object in a location, it adds this to the current set of training instances. ObjectEval will then search the Web and use the resulting webpages as additional training examples that relate the object to the observed location.

Table 5.2 shows the results of different approaches that have been used to find objects. The baseline only uses the distance and interaction terms of Equation 5.2 to greedily generate the next location to visit and uses no semantic information about the environment. ObjectEval maximizes the expected utility Equation 5.2 in both offline or interactive modes.

Approach	**Visited locations**		**Distance**		**Interactions**	
	Mean	Standard Error	Mean	Standard Error	Mean	Standard Error
Baseline	35.8	6.1	69.6	7.2	71.5	12.3
ObjectEval (offline)	14.3	4.3	33.9	4.6	28.7	8.6
ObjectEval (interactive)	10.2	3.8	32.5	4.4	20.5	7.7

Table 5.2: Average and standard error for the number of visited locations, distance and number of interactions for ObjectEval and baseline approaches. The baseline uses only the terms for interaction I and distance D from Equation 5.2. ObjectEval (offline) uses batch training and ObjectEval (interactive) is given no training data, but instead uses the presence of objects in locations to update the probability of a location given the object as it performs a search (as from Equation 5.2).

There is a clear downward trend in the number of visited locations and the number of interactions for ObjectEval when compared with this baseline, indicating that the system is learning about the physical environment. Surprisingly, the interactive mode of ObjectEval achieves better results then the offline version of ObjectEval. Since the training data from

Mechanical Turk can be different from the objects and the locations that are found by the robot, the interactive version of ObjectEval may have an advantage since it learns the locations of objects directly in the test environment. The offline version starts with a biased set of data (obtained from Mechanical Turk) that may not accurately reflect the real-world. For example, people from Mechanical Turk have annotated "cup" or "glasses" as examples of objects that can be found in a bathroom. However, in our office environment, these objects are expected to be found in offices. By training on these examples, the offline version of ObjectEval would be biased toward finding these objects in the bathroom, whereas the interactive version does not have this problem because it only uses training data about objects in the environment.

Although the number of visited locations in Table 5.2 may seem high, the interactive version of ObjectEval finds 80% of the objects within five locations or less, whereas the baseline finds only 39% in the same five locations. One reason that this term is high is because of a high penalty for choosing the wrong location. For example, if the robot incorrectly classifies "soap" as being in an "office", it will have to search an order of magnitude more locations because the environment contains hundreds of offices, whereas it only contains a few bathrooms.

Finally, we have profiled the number of locations visited before finding an object. Figure 5.7 shows the result when a search for 20 objects is repeated 5 times starting from different initial location to obtain 100 runs. The figure shows that ObjectEval, after having gathered only a few facts, has quickly learned to execute efficient plans to find objects when compared with the baseline approach.

5.3.3 Robot Experiments

We have demonstrated the ability of ObjectEval to find and deliver an object on our mobile office assistant robot. In order to show this, ObjectEval is integrated into KnoWDiaL [Perera et al., 2015], an approach for learning task-relevant environmental knowledge by interacting with humans or processing information on the Web. OpenEval is used as one of the main five components of the KnoWDiaL approach, which namely are: a frame-semantic parser, a probabilistic grounding model, a knowledge base, a web-based predicate evalua-

Figure 5.7: The number of locations visited by the robot before finding the query object for the interactive mode of ObjectEval (red line) and the baseline (green line). The data is sorted by the number of visited locations per simulation run.

tor which uses OpenEval, and a dialog manager.

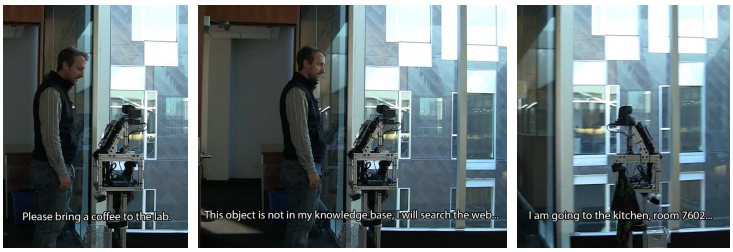
The interaction of a user with the KnoWDiaL system is as follows. The user first provides a spoken command, e.g., “Get me a coffee”. Then frame-semantic parser processes the input command and extracts a set of candidate frames containing slots for phrases that are referring to action types (e.g., get), and phrases that are referring to action parameters (e.g., the name of object: coffee). The probabilistic grounding model then maps the extracted set of frames to *referents* in the knowledge base. For example, it maps the word “get” to action “transportObject”. In some cases, the information of some of the frames is not available. For example in the command “get me a coffee”, it is not clear where the object should be delivered or where the robot should find the object “coffee”. In these cases, the dial manager fills the missing fields via dialog or via querying OpenEval (e.g., by querying OpenEval to determine if “coffee” is likely to be found in the “kitchen”). Finally, the dialog manager gets the task confirmation from the human, executes the task, and updates the knowledge base.

The KnoWDiaL approach is implemented on the CoBot robot, which enables it to interact with humans through the dialog interface and autonomously navigate in office buildings. OpenEval improves the efficiency of the KnoWDiaL approach, in terms of required user interactions, by retrieving part of the knowledge from the Web instead of interacting with humans. By querying OpenEval (e.g., asking whether the claim "coffee is likely to be found in a kitchen" is correct), the robot is able to avoid asking the human about some of the common sense information that it needs in order to perform a task.

KnoWDiaL system is used by ten different people, asking the robot to deliver an object or to collect an object from a location and deliver it to another location. When the robot is asked to pick up an object, the location that the object should be picked up can be either explicitly specified by the input command (e.g., bring me coffee from the kitchen), or not be specified in the command (e.g., bring me coffee). In the case that the location is not specified, the robot either finds the location in its knowledge base or queries OpenEval. The baseline used in the experiment simply asks a question from the user for each of the parameters needed to execute an action. The experiment is started by an empty knowledge base, and the robot extends its knowledge base as it interacts with humans. A total of 91 speech-commands are received by the robot. The details of the experiments is published in [Perera et al., 2015].

Among all 31 commands that the robot receives to pick an object from a specific location, 19 times the location of the object could not be extracted either from the command or from the knowledge base. In this case, OpenEval is queried to extract the most likely location for an object. OpenEval has been able to come up with a correct object location 11 times. Among these cases, 2 times OpenEval returns a location with a score that is significantly higher than other candidate locations, 7 times two of the locations have a high-probability score, and 2 times three of the locations have a high-probability score. In case more than one location has a high probability score, then the robot clarifies the correct location by interacting with a human.

Figure 5.8 shows an example of interaction between a user and CoBot. The human asks the CoBot to bring the object "coffee" without specifying the location where the object should be picked up from. Since the object is not specified in the input command, the CoBot queries OpenEval to find out the probability of object "coffee" to be found in



User:	Please bring a coffee to the lab.
CoBot:	The object is not in my Knowledge Base, I will search the web using OpenEval to see if the object is in location “kitchen”, “office”, “meeting room” or “printer room”.
CoBot:	OpenEval returns high probability for object “coffee” in location “office” and also for object “coffee” in location “kitchen”. Which location do you think is the best?
User:	Kitchen.
CoBot:	I will collect the object “coffee” and deliver it at the lab. I am going to the kitchen, room 7602.

Figure 5.8: An example of interaction between a human and CoBot. CoBot queries OpenEval in order to retrieve the information that it needs to execute the required task (e.g., determining where it is likely to find object “coffee”).

either “kitchen”, “office”, “meeting room”, and “printer room”. OpenEval then returns high probability for object “coffee” to be found in both locations “office” and “kitchen”. Since both of these locations have high probability, CoBot disambiguates the location that the object should be picked up from by asking a human. Finally, the CoBot goes to the “kitchen” to get the object “coffee”.

5.4 Summary

In this chapter, we presented an approach, called ObjectEval, which is able to find and deliver objects in real-world environments. We showed that our system learns to query

the Web to evaluate the probability of physical background knowledge that relates objects and locations. In addition, we present an approach for finding and delivering objects that integrates information about object locations from the Web, and interactively learns about the physical environment by getting feedback from humans. We showed promising results over a baseline approach and have demonstrated our system on a mobile robot navigating in an indoor environment.

Chapter 6

Iterative Query-Based Open World Planning

In the previous chapter, we showed how a real mobile robot (CoBot) can iteratively query the OpenEval approach to retrieve the information that it needs to execute the required task. In this chapter, we contribute an automated planning approach that actively queries the open Web using the OpenEval approach to add needed knowledge until the planner solves the planning problem.

We begin in Section 6.1 by motivating the open world planning problem, and explaining how to relax the assumption that all the knowledge, in terms of predicate instances, is provided as an input to the planner. We explain the details of our approach, Open World Planner (OWP), in Section 6.2. Section 6.3 presents the detail of our experiments to evaluate the OWP approach. Finally, Section 6.4 presents the summary of this chapter.

6.1 Introduction

Most of the current planning techniques assume that the planner has complete information about the problem. For example, consider a trip planner that plans a trip from an initial city to a destination city, and also finds hotels and attractions under a set of constraints. This planner would require the names of the cities, the corresponding hotels and attractions and properties. For the trip planner to be able to plan a trip from any arbitrary city, it would probably require several thousands of instances of predicates to define the complete planning problem. The knowledge of the planner would also need to be updated since instances and properties can change over time.

Providing all of the relevant knowledge to the planner seems to be impractical. Manually entering the knowledge is infeasible in practice due to the effort that it requires to gather the information and to keep it updated. Identifying and extracting upfront all of the relevant knowledge, including instances and relations, for a real planning problem is also not trivial. Finally, if one were able to retrieve all of the relevant information (e.g., from the Web), most of the current planners would, in practice, not be able to use such large amounts of extracted knowledge, since the search space would drastically increase.

These challenges have been addressed by considering the planning problem with incomplete information, where the knowledge needed for planning is retrieved during plan execution. Most of the current planning techniques that deal with incomplete knowledge are designed for gathering information during plan execution by inserting sensing actions in the plan [Etzioni et al., 1992; Golden et al., 1996; Weld et al., 1998; chi Tuan et al., 2004]. The output of these planning techniques is a plan conditioned on the information that will be gathered by the sensing actions during the plan execution. The missing information is sporadic and can also be handled also in a mixed iterative manner by asking users to fill it in or through selecting planning choices [Talamadupula et al., 2013]. Other techniques plan to gather information from Web services about an assumed incomplete initial state [Kuter et al., 2004; Sohrabi et al., 2006]. The main assumption behind these techniques is that all of the information relevant to the planning problem is accessible to the planner through a set of Web services.

In this chapter, we contribute a new general automated approach for planning with

incomplete information, which we call Open World Planner (OWP). The OWP estimates the missing relevant knowledge, automatically queries the open Web for it and assigns confidence given its uncertainty, translates the information for use by a classical planner while reasoning about its uncertainty, and repeats this process iteratively until a solution is found or a termination criterion is reached. Briefly comparing OWP with other work on planning with incomplete information, (i) OWP does not rely on specific knowledge being provided by a Web service, (ii) does not add sensing or mixed-initiative actions in the plan, (iii) and also considers the uncertainty that exists in the retrieved information. OWP could be integrated with any of the other techniques, but its contribution at this time is its automated functionality.

OWP receives as input a planning domain and a problem expressed in the PDDL language. The planner also gets a few seed examples for the predicates that the planner may need in order to gather information during the planning, e.g., a few names of cities as predicate instances of the predicate *(city ?x)*. The input seed examples are used by OWP to learn how to extract new instances of each predicate. OWP then actively queries the open Web using the OpenEval approach to acquire new predicate instances while solving the planning problem. To determine which predicate instances need to be acquired, OWP first estimates the relevant knowledge to the initial state of the planning problem (knowledge in terms of predicate instances), and effectively retrieves this knowledge by querying OpenEval. Our planner then iteratively adds the retrieved knowledge to the planning problem based on a knowledge-confidence value computed from the retrieved information and proceeds to search for a solution to the planning problem. If it fails to solve the problem given the extracted knowledge, it iteratively continues the process of adding new knowledge and solving the problem, until it finds a solution or reaches its termination criterion, e.g., time or number of iterations.

We illustrate OWP within the trip planning domain. We show that OWP is able to plan a trip for a variety of different problems when no knowledge is given as input to the planner. We run our experiments on three different types of trip planning domains with different levels of difficulties. For each of these domains, we use 10 different problem instances with different start and destination cities. OWP proceeds to solve the problems using the open Web. Our experiments show that our approach is able to decrease the

number of queries sent to the Web by a factor of at least two orders of magnitude compared to the baseline approach.

6.2 OpenWorld Planner

Algorithm 9 shows the pseudocode of our Open World Planner (OWP). OWP consists of three main parts, namely: constructing a knowledge graph and extracting constraints (Line 5-6 in Algorithm 9), instantiating the constraints (Lines 7-12), querying the Web and calling Downward planner [Helmert, 2006a] (Lines 13-22).

In the rest of the chapter, we exemplify our approach using a very simplified version of a trip-planning domain which consists of booking a hotel, traveling to a city near the start city, and visiting a museum. Part of a PDDL-like encoding of our domain is shown in Figure 6.1. In this domain, seed examples to train OWP are examples of cities for the predicate *city*, as well as seed examples for predicates *city-close-city, hotel, hotel-in-city, museum,* and *museum-close-to-hotel*.

```
Init: (in-city Seattle) (city Seattle)
Goal: (exists (?x)(not (=?x Seattle)(visited-city ?x))
Operator Book-Hotel
 Pre:(city ?x), (hotel ?y),(hotel-in-city ?y ?x)
 Eff:(booked-hotel ?y ?x)
Operator Travel-To-City
 Pre:(city ?x),(city ?y),(in-city ?x)
  (city-close-city ?x ?y),(booked-hotel ?z ?y)
 Eff:(not(in-city ?x)),(in-city ?y),(traveled-to-city ?y)
Operator Visit-Attraction-In-City
 Pre:(in-city ?x),(museum ?y),(booked-hotel ?z ?x),
  (museum-close-to-hotel ?z ?y),(traveled-to-city ?x)
 Eff:(visited-city ?x)
```

Figure 6.1: Simplified version of trip planning problem (not the actual PDDL format).

Algorithm 9 Open World Planner

Require: $Domain, Problem \leftarrow$ Definition of planning domain and problem in PDDL; $S \leftarrow$ Seed examples for static predicates

Ensure: Return a valid $plan$ or return false if no plan has been found

Function OpenWorld Planner
$Thresh \leftarrow 1$
$S \leftarrow$ Predicate instances in the initial state of the input problem
while no plan has been found **do**
  $G \leftarrow$ Construct knowledge graph starting from initial state $\mathcal{I}$
  $C \leftarrow$ Extract a set of constraints C from G
  Instances $\leftarrow \{\}$
  Sort the constraints in C by their *priority* value
  for all constraint $c \in C$ **do**
    Crawl objects for constraint c from the Web
    Instances $\leftarrow$ *Instances* + instantiate predicates using constraint c
  end for
  for all instance $i \in$ *Instances* **do**
    Call OpenEval to validate correctness of i
    if i is correct and *confidence(i)* $\geq$ *Thresh* **then**
      Add i to $\mathcal{I}$
    end if
  end for
  if no new instance is added to $\mathcal{I}$, **then** break
  $plan \leftarrow$Call Downward to solve planning problem
  if plan is not empty, **then** return $plan$
  Thresh $\leftarrow$ *Thresh* $- \epsilon$
end while
return False

6.2.1 Definitions and Notations

Here we define the notation that we use in this chapter. A *PDDL task* is denoted by a 5-tuple $\mathbf{P} = \langle \mathcal{L}, \mathcal{I}, \mathcal{G}, \mathcal{X}, \mathcal{A} \rangle$, where $\mathcal{L}$ consists of *objects* (constant symbols), *predicates* (relation between objects), and a set of *variable symbols*. The initial state $\mathcal{I}$ is a conjunction of ground atoms over objects. The goal state $\mathcal{G}$ is the conjunction of positive and negative *literals*. A *literal* is either an atom or negation of an atom. A *predicate* can be ei-

ther partially- or fully-grounded (i.e. fully-instantiated). A literal is also called a *predicate instance*, i.e. where all the arguments of a predicate are grounded. $\mathcal{X}$ is a set of *schematic axioms*. For simplicity and without loss of generality, we assume that $\mathcal{X}$ is empty throughout the paper. $\mathcal{A}$ is a set of *action schemas* which implicitly define a set of *ground actions*. Each action schema consists of an action name, list of variables used in the action schema, a precondition, and an effect. Precondition and effect of an action are defined by a logical sentence consisting of conjunctions of literals. A *ground action* is applicable in state s if its preconditions are satisfied by s. Similarly, an action schema is applicable in state s, if there is at least one substitution for its variables that makes the obtained ground action applicable in s.

In our notations, we also differentiate between *static* and *dynamic* predicates. A *static* predicate is a predicate whose value of instances (i.e., literals) in the initial state can not be changed by the effect of any of the actions. A predicate is called *dynamic* if its instances can be deleted or added by some of the actions. This would help us to differentiate between the literals that are added by retrieving and processing knowledge from the Web (called *static literals*) and the literals that are added by the effect of actions (called *dynamic literals*).

6.2.2 Constructing Knowledge Graph

Starting from the initial state, OWP first estimates the relevant knowledge to the initial state of the planning problem. Our objective is to estimate the positive literals (ground atoms) that are most likely to be used during the search. For example, if we are finding a trip between two cities in the United States, adding information about the cities in Europe is not only irrelevant to the solution but also increases the time complexity (e.g., takes longer to search) and space complexity (e.g., more grounded operators) of solving the planning problem.

To estimate the relevant knowledge to the initial state of the planning problem, we first construct a *knowledge graph* (Line 5 in Algorithm 9). A knowledge graph is a compact structure that implicitly encodes all of the literals that are likely to be used during planning. A knowledge graph is also used to prioritize the ground atoms that should be added to the

planning problem in order to solve the problem. Note that the structure and some of the properties of our knowledge-graph is somewhat similar to the Planning Graph [Blum and Furst, 1997].

Definition 1. Knowledge graph *is a layered graph where its vertices are partitioned into a sequence of layers and its edges connect successive layers. Each layer of knowledge graph contains a set of partially- or fully-grounded predicates and each edge corresponds to one of the ground actions in the planning problem. The first level of the graph contains all of the atoms in the initial state $\mathcal{I}$.*

The knowledge graph is constructed layer by layer, starting from the first layer. We represent each layer of the graph by L_i where i is the level number. Each layer L_i contains two sets: L_i^f and L_i^p. L_i^f is a set of literals and L_i^p is a set of partially or fully-grounded predicates also called *constraints*.

Definition 2. Constraints *are defined by a set of fully or partially-grounded predicates present in layer i of the knowledge graph (denoted by L_i^p). Each predicate in L_i^p defines a constraint on the values that each argument/variable of a predicate can potentially take.*

For example, a constraint such as $\langle$*(hotel-in-city ?x ?y), ?y=Seattle*$\rangle$ binds the predicate *(hotel-in-city ?x ?y)* to only take instances that have city *Seattle* as the second argument (i.e., all the hotels in *Seattle*). This constraint can be simply written as *(hotel-in-city ?x Seattle)*.

6.2.3 Layer-by-layer construction of knowledge graph:

First layer (L_0): For the first layer of the knowledge graph, L_0^f contains all the ground atoms in the initial state. L_0^p is set to be empty.

Middle layers (L_i): Assume that A_i is the set of all the ground actions whose preconditions are a subset of L_i^f, i.e. all the actions that are applicable to L_i^f . L_{i+1}^f is constructed by the union of the effects of all actions in A_i. Note that L_i^f could potentially have both an atom and its negation. We also allow "no-op actions", so every literal that appears in L_i^f may also appear in L_{i+1}^f.

For each layer, we also create a set of constraints L^p_{i+1} that are later used to identify and bound new atoms that should be added to the planning problem. Each constraint consists of a partial or full assignment to the arguments of a predicate. The constraints are created as follows: for each action $a \in \mathcal{A}$, we check all of the subsets of L^f_i, denoted by s, that partially or fully satisfy preconditions of action a. Let's assume that a' is the action obtained by applying s on a. Depending on s, the precondition of a' can be either fully or partially instantiated. For each predicate p in the precondition of a' that is either partially-instantiated or is not part of the L^f_i, we add p as one of the constraints in L^p_{i+1}. For each predicate/constraint p, we also assign an integer value $N(p)$ which indicates the number of different substitutions s that can result in p. Constraint p tells us that in order to be able to apply action a', the planner requires to have instance(s) of p. Adding p, helps OWP to decide what predicate instances need to be added in in order to solve the planning problem.

Stopping condition: The iteration of creating the knowledge graph continues until $L^f_i = L^f_{i+1}$.

Given a planning problem with n objects, $|\mathcal{I}|$ number of atoms in the initial state, $|\mathcal{A}|$ number of action schemas where each action schema has a constant number of parameters and at most m number of distinct predicates in its effect, it can be easily shown that the size of L^f, L^p, and the time to create a knowledge graph with $|L|$ number of levels, are polynomial in terms of n, $|\mathcal{I}|$, $|\mathcal{A}|$, m, and $|L|$. The proof is similar to the proof of theorem one in [Blum and Furst, 1997].

6.2.4 Example of knowledge graph

Figure 6.2 shows part of the knowledge graph that is built for our trip-planning example. The first level of the knowledge graph, L_0, contains two sets of literals: those that exist in the initial state of the planning problem and those that are retrieved from the Web in the previous iterations of our algorithm. We explain in the next section how these literals are retrieved. L^p_0 is empty and is not shown in the figure, therefore $L_0 = L^f_0$. The second layer of the graph contains a set of literals (L^f_1) and a set of constraints represented by partially or fully-instantiated predicates (L^p_1).

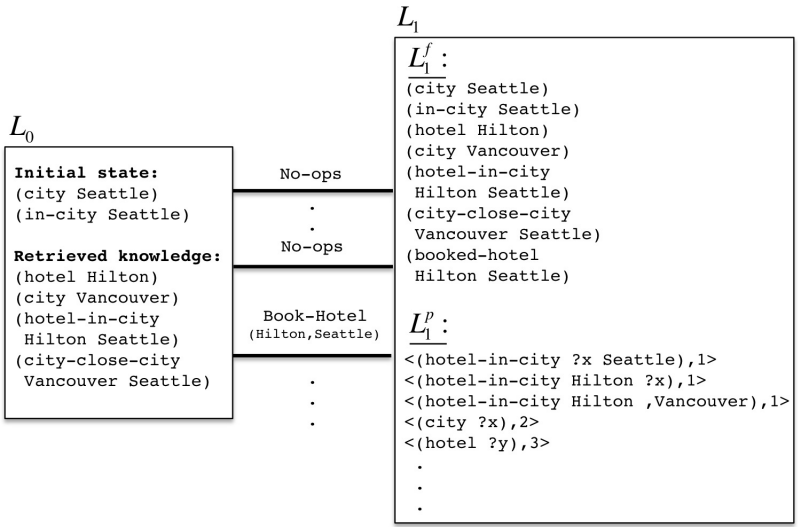


Figure 6.2: Part of the knowledge graph built for the simplified version of the trip-planning domain.

Figure 6.3 shows an example of constraints that are extracted for action *Book-Hotel*. For simplification, assume that we only have three atoms in the layer L_0 of the knowledge graph: *(city Seattle), (city Vancouver), (in-city Seattle)*. There are two substitutions of variables that partially satisfy the precondition of action *Book-Hotel*, one when $?x = Seattle$ and the other one when $?x = Vancouver$. Each of these substitutions binds values of variable *?x*. These two bindings give us three different constraints. Note that we also consider empty substitution which does not bind values of any of the variables in the precondition of an action.

6.2.5 Prioritizing constraints

OWP next assigns a priority value to each constraint. The **priority value** is calculated by multiplying: (i) *effectiveness* value of each constraint and (ii) the *confidence* value of OWP on the correctness of a constraint. The effectiveness value of a constraint approximates the likelihood of the predicate instances, determined by this constraint, to be used during the search while solving the planning problem.

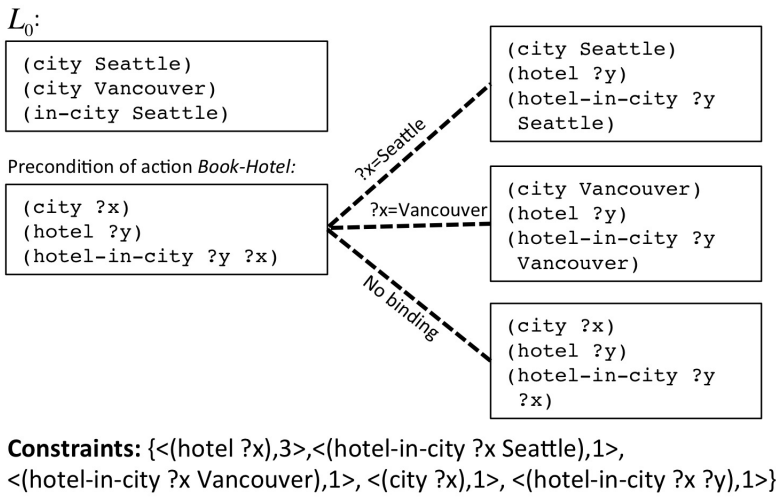


Figure 6.3: Examples of constraints that are extracted for action *Book-Hotel*.

Definition 3. *The* **effectiveness value** *of a constraint c is defined as:*

$$e(c) = \sum_{i=0}^{|L|-1} \mathbf{1}_{L_i}(c)\Big(w_1 \frac{|L|-i}{|L|} + w_2 \frac{N(c)}{N_{max}} + w_3 R(c)\Big) \tag{6.1}$$

where $|L|$ is the number of layers in the knowledge graph, $\mathbf{1}_{L_i}(c)$ is an indicator function which returns 1 if c is in layer L_i of the graph and zero otherwise, w_i is a constant, $N(c)$ is the number of times that constraint c appears in the level L_i, N_{max} is the maximum value of $N(c)$ over all different constraints in the different levels of the graph, and $R(c)$ is the ratio of the number of arguments of c that are instantiated to the total number of its arguments.

Intuitively, the effectiveness value of a constraint depends on three factors: (i) how early a constraint appears in the knowledge graph; if a constraint appears early, it bounds the knowledge that we may need in the future layers of the graph, (ii) the frequency of a constraint that indicates how many different actions require instances of the constraint, and (iii) the ratio of the arguments that are instantiated to the total number of arguments. The weights w_i indicate the importance of each factor and are set manually in our experiments.

The confidence value of a constraint is calculated based on how confident OWP is that an extracted constraint is correct. Assigning a confidence value to each constraint is an important step since some of the literals in the knowledge graph might have been extracted incorrectly (e.g., due to an error in retrieving knowledge from the Web). These incorrect literals may then create incorrect constraints. For example, assume that OWP adds *(city Hilton)* as a literal to the knowledge graph which can potentially create an incorrect constraint such as *(hotel-in-city ?y Hilton)* (i.e., hotels *?y* that are in city *Hilton*). To reduce the error of extracting incorrect constraints, a confidence value is calculated for each constraint. To define the confidence value of a constraint, we first need to define the confidence value for each literal that exists in L_i^f.

Definition 4. *The* **confidence value of a literal** $l \in L_i^f$*, denoted by* $\bar{c}(l)$*, is defined as:*

$$\bar{c}(l) = \begin{cases} 1 & l \in \mathcal{I} \\ p & l \textit{ is retrieved from the Web} \\ \max\limits_{\forall a \in A \& l \in \textit{eff}(a)} \bar{c}(\textit{pre}(a)) & l \textit{ is a dynamic literal} \end{cases} \tag{6.2}$$

where p *is the confidence value of retrieving* l *from the Web,* A *is the set of all ground actions, pre(*a*) is the precondition of action* a*, and eff(*a*) is the effect of* a*.*

If literal l is part of the initial state, then its confidence value is equal to 1. If l is retrieved from the Web, then its confidence value is equal to the confidence value of OWP on extracting l from the Web. If l is created as an effect of some ground actions applied on the previous layer of the knowledge graph, then the confidence value of l is calculated as follows. For each ground action a that results in l as its effect, OWP first calculates the confidence value of the precondition of a, denoted by $\bar{c}(\textit{pre}(a))$, and then sets $\bar{c}(l)$ equal to the maximum of the confidence values over all the actions a. A precondition of each ground action a is a logical sentence that consists of a set of literals whose confidence value is known (calculated in the previous layer of the knowledge graph). To calculate the confidence value of a logical sentence, we use the same operations that are used in fuzzy logic [Lee, 1972], for instance, logical operators *and* and *or* are respectively represented by *min* and *max* operators.

Proposition 1. *If* $0 \leq p \leq 1$*, then for all literals* l*,* $0 \leq \bar{c}(l) \leq 1$*.*

The definition of the confidence value of a literal helps us to define the the confidence value of a constraint.

Definition 5. *The* **confidence value of a constraint** $c \in L_i^p$*, denoted by* $\bar{c}(c)$*, is defined as:*

$$\bar{c}(c) = \max_{a \in A(c)} \bar{c}(\textit{pre}(a)) \tag{6.3}$$

where $A(c)$ *is a set of all actions that created constraint* c*.*

To calculate the confidence value of constraint c, we iterate over all the actions that created constraint c, and then calculate the confidence value of their preconditions, denoted by $\bar{c}(\text{pre}(a))$. Since a might be partially-instantiated, $\bar{c}(\text{pre}(a))$ is calculated only for the predicates in $pre(a)$ that are fully-instantiated.

6.2.6 Instantiating Constraints

Given a set of constraints, OWP first sorts the constraints based on their priority value and then instantiates each constraint to a set of literals (Lines 8-12 in Algorithm 9). To instantiate constraint c, OWP first crawls a set of objects from the Web by searching the English name of the predicate in c and its arguments on Google and crawling the phrases that appear in the returned webpages. We use ReVerb [Fader et al., 2011a], which is an information extraction tool that automatically extracts unary and binary relationships from English sentences, to extract phrases from the returned webpages. For example, given the constraint *hotel-in-city(?x Seattle)*, OWP searches query *"hotel in city Seattle"* on Google and extracts phrases from the returned webpages. To limit the number of retrieved objects, in each iteration of Algorithm 9 (Line 4), OWP only extracts phrases from one of the webpages returned by Google.

After extracting objects relevant to a constraint, OWP goes through all of the variables in the constraints that are not instantiated and substitutes each variable with one of the objects extracted for the constraint. To decide the number of instances that should be instantiated for each constraint, OWP first divides the priority value of the constraint by the sum of the priority values of all the constraints, and then multiplies it by the total number of instances that should be instantiated.

6.2.7 Querying OpenEval and Calling Downward Planner

Given all the candidate predicate instances that are obtained by instantiating the constraints, OWP then queries the Web using the OpenEval information extraction technique to validate the correctness of each candidate predicate instance (Lines 13-18 in Algorithm 9). Given the output of OpenEval for each queried instance, OWP then adds to the planning problem all the predicate instances that have a confidence value greater than threshold *Thresh* and are evaluated by OpenEval to be correct (Lines 13-18 in Algorithm 9). Fast Downward [Helmert, 2006a] is then called to find a solution to the planning problem (Line 20). If no solution has been found, OWP iteratively continues the process of extracting and instantiating constraints until a solution is found. Note that our algorithm starts by setting the value of threshold $Thresh$ to one and decreases its value in each iteration.

6.3 Experimental Results

Our experiments investigate how well Open World Planner would perform in solving a set of trip-planning problems, where we assume that no information is given to the planner (except input seed examples). We run our experiments on three different types of trip planning domains with different levels of difficulties. The difficulty of these domains is determined by the number of actions that are defined in the domain and the number of predicates used in the precondition of each action. In the first domain, our goal is to travel to a destination city and book a hotel in the city. In the second domain, we consider planning a trip to a destination city where the planner should also find a hotel and two attractions. In the third domain, our goal is to find a trip starting from an initial city to a destination city where another city is visited in between (the planner should find two attractions and one hotel in each city). For each of these domains, we use 10 different problem instances with different start and destination cities. The only objects that are provided to the planner are the names of the initial city and the goal city. The details of the domains and the problems used in our experiments are published in Appendix A.

Table 6.1 shows the average number of queries sent to the Web (OpenEval) and the

Problem Set (10 problems per set)	**No. Queries (Full-Instantiation)**		**No. Queries (OWP)**		**Plan Accuracy (OWP)**	
	Mean	SE	Mean	SE	Mean	SE
One City	7078.2	706.2	28.6	4.8	80%	8%
One City - Two Attractions	53189.3	6661.6	452	72.9	68%	7%
Two Cities - Four Attractions	92483.6	6772.7	655.7	40.4	56%	4%

Table 6.1: Total number of queries sent to the Web (OpenEval) using Open World Planner and the baseline approach that instantiates all of the predicate instances. The table also shows the accuracy of plans found by OWP.

accuracy of plans found by OWP. We are reporting the number of queries sent to the Web because retrieving the information from the Web is the most time consuming part of the planning process. The first column shows the type of the planning problem. The second two columns show the average and the standard error for the number of queries that are sent to the Web for the baseline approach. The baseline approach (called *full-instantiation*) is built by iterating over all the possible ways of instantiating predicates in the domain, calling OpenEval to verify each predicate instance, and adding the instance to the planning problem if it is evaluated by OpenEval to be correct (with the confidence value greater than 0.5). To instantiate the predicates, we use the same objects that are retrieved by OWP from the Web. The next two columns show the number of queries that are sent to the Web by OWP. The result shows that Open World Planner on average decreases the number of queries that are sent to OpenEval by at least by a factor of at least two orders of magnitude compared to the full-instantiation approach. For these experiments, we set $w_1 = 1$, $w_2 = 2$, and $w_3 = 5$.

Since the output of OpenEval is not completely accurate, some of the steps found by our planner may be incorrect. For example, we observed that OpenEval returns *Science World* as a museum located in *Seattle* when it is actually located in *Vancouver*. Table 6.1 also shows the result for the accuracy of plans found by OWP. This result is obtained by manually checking each step of the plans found by the planners, and counting the number of the steps that are correct. Figure 6.4 shows one of the plans that is automatically found

by OWP.

```
Book-Hotel: Omni Hotel, Austin
Travel-To-City: Boston, Austin
Visit-Museum: Mexic-Arte, Austin
Visit-Museum: Blanton Museum of Art, Austin
Book-Hotel: Omni Chicago, Chicago
Travel-To-City: Austin, Chicago
Visit-Museum: Omni Chicago, Chicago
Visit-Museum: Museum Science, Chicago
```

Figure 6.4: An example of a plan found by Open World Planner for traveling from *Boston* to *Chicago* while visiting an arbitrary city (*Austin*) in between.

6.4 Summary

In this chapter, we presented a new planning approach, called Open World Planner, that actively queries the open Web to acquire instant knowledge about the planning problem. We presented the details of our technique that estimates the relevant knowledge to the initial state of a planning problem by constructing the *knowledge graph* and extracting a set of constraints. We showed promising results over a baseline approach and have shown that our technique is able to significantly decrease the amount of time required to solve a planning problem.

Chapter 7

Related Work

Our work in this thesis leverages and extends some of the ideas explored by previous work in open information extraction, machine reading, trust, searching for objects using mobile robots, budget-sensitive classification, and the integration of robots using the Web. In this section, we briefly describe salient research from each of these fields.

7.1 Use of Co-Occurrence Statistics in Information Extraction

Co-occurrence statistics computed from a collection of documents such as Web corpus has been recently widely used for IE. Turney et al. were among the first who realized that search engines can be used to compute Web-scale co-occurrence statistics [Turney, 2001, 2002; Turney and Littman, 2003]. They presented an unsupervised technique that uses search engine hit counts to measure the similarity of pairs of words. They used the Pointwise Mutual Information (PMI) technique to measure the semantic similarity between two words. Other researchers also have used PMI to validate information extraction [Soderl et al., July 2004] or to validate a candidate answer in a question answering system [Magnini et al., 2002]. Soderland et al., for instance, showed different variations of the PMI technique that was used to assign a correctness probability to the extracted

facts [Soderl et al., July 2004]. To compute the PMI score, they assumed that a set of *discriminator phrases*, such as "cities of" for predicate *city*, are given to the system. These discriminator phrases can be obtained by IE systems such as KnowItAll [Etzioni et al., 2004].

Magnini et al. proposed an answer validation approach that uses the redundancy of Web information to validate a candidate answer for a question [Magnini et al., 2002]. Given an input question and a candidate answer, their technique first extracts a set of keywords from the input question and the candidate answer. An *answer validity score* is then obtained by using the search engine hit counts for the extracted keywords. Other researchers [Brill et al., 2001] have also studied the usage of redundancy of information on the Web for answer validation.

Co-occurrence statistics of words has also been also used to classify a hypothesis relation instance. Cimiano and Staab developed a method to use search engine hit counts to classify a hypothesis relation instance [Cimiano and Staab, 2004]. Their system first iterates through a set of entities to be classified. It then generates a set of patterns for each concept that exists in the ontology. These patterns are then searched in Google and the hit counts for all the results are summed up. The classification is done based on the number of Google counts for all pattern instances. Similar to this work, Navok et al. also showed that co-occurrence statistics obtained from Web data can be used to to find names for noun-noun relations [Nakov and Hearst, 2005b,a,c].

The performance of all of the above techniques depends on the accuracy of the search engine hit counts. However, the search engine hit counts at best are only a crude estimate of the number of matching documents. It has been shown that hit count estimates are not accurate [Rousseau, 1999; Anagnostopoulos et al., 2006; Uyar, 2009]. In fact, researchers have shown that the hit counts change daily [Rousseau, 1999], are unreliable especially for disjunctions [Anagnostopoulos et al., 2006], and their accuracy is reduced almost by the half when going from a one word to two words query [Uyar, 2009]. Our approach *does not rely* on the search hit count. Instead, it uses *the content* of the webpages that are returned by the search engines for the evaluation.

We are not the first to not to rely on the search engine hit counts; Downey et al. developed a combinatorial "balls-and-urns" model (Urns model) that computes the impact

of redundancy on the probability of the correctness [Downey et al., 2005] . Their model is based on the fact that repeatedly obtaining the same extraction from different independent sources increases the correctness probability of the extracted instance. They showed that this model is superior to the PMI method based on both accuracy and time. However, the main drawback of their system is that the Urns model can not be used *independently* to evaluate correctness of a predicate instance and it requires the output of an IE system such as *KnowItAll*.

7.2 Extracting Factual Information from Unstructured Web Text

The main goal of machine reading (i.e., learning by reading) is to automatically extract knowledge from an unstructured text corpus (e.g., Web) and represent the knowledge in a structured form that can be used by machines. The main challenge is to scale the machine reading techniques to a large corpus such as the Web. The Web contains texts with different levels of heterogeneity, vary in subject matters (health vs. food), writing style (blog vs. scientific papers), etc. which make the texts difficult to process. As a result, machine reading techniques must rely on indirect supervision, learning, and improving its performance as it reads the text [Poon et al., 2010].

The main general approach for indirect supervision in IE is to use meta knowledge about the domains and extract information based on the data redundancy. Etzioni and colleagues were among the first researchers who followed this approach and developed KnowItAll, a web-scaled IE system that is based on indirect supervision to extract lists of instances of a given category/relation [Etzioni et al., 2004]. KnowItAll is based on an observation that there exists general patterns that can be used to extract relation instances. KnowItAll combines hyponym patterns [Hearst, 1992] and learns new patterns for instances of a relation/category. These patterns are then used to identify and extract named-entities. In addition to the text patterns, KnowItAll also uses wrapper algorithms [Crescenzi and Mecca, 2004] to extract information from structured contents such as tables in the websites. Downey et al. showed that the accuracy of KnowItAll can be

improved by post-processing the extracted instances using a combinatorial model based on the redundancy of information on the Web [Downey et al., 2010, 2005].

Etzioni's group at the University of Washington continued their work on KnowItAll and later introduced TextRuuner [Etzioni et al., 2008; Poon et al., 2010], a highly scalable IE system, that is shown to achieve an error reduction of 33% compared to KnowItAll. Similar to these works, Carlson et al. [Carlson et al., 2010b,c] developed NELL, a semi-supervised learning technique that has been developed as part of the Read the Web (RTW) project at Carnegie Mellon University. NELL extracts structured information from unstructured webpages. The input of NELL is an initial ontology (e.g. hundreds of predicates with one or two arguments) and 10 to 15 seed examples for each predicate that is defined in its ontology. Given this input, NELL extracts new instances of predicates from a collection of 500 million webpages (the ClueWeb09 corpus [Callan and Hovy, 2009]). NELL first extracts a set of candidate instances and then filters those that have a low confidence value.

Etzioni et al. [Etzioni et al., 2011; Fader et al., 2011b; Mausam et al., 2012] later developed ReVerb, a general verb-based relation extractor, that uses a set of generic syntactic and lexical constraints to identify and extract relation instances. They showed that approximately 85% of the binary verbal relation phrases in a sample of Web sentences satisfy their constraints. ReVerb addressed two limitations of previous Open IE systems: *incoherent extractions* and *uninformative extractions*. Incoherent extractions refers to the cases when the IE system extracts a relation that doesn't have a meaningful interpretation and uninformative extractions refers to the cases when the IE system omits critical information in the relation extraction.

One of the main advantages of approaches like ReVerb [Fader et al., 2011b], TextRunner [Poon et al., 2010], WOE^{pos} [Wu and Weld, 2010], and R2A2 [Etzioni et al., 2011] is their efficiency since they are using only shallow syntactic parsing (chunking and part-of-speech tagging) in their extractor. However, the efficiency is obtained in most of these systems at the cost of accuracy (lower precision at higher points of recall). To achieve better precision and recall, approaches such as OLLIE [Mausam et al., 2012], Wanderlust [Akbik and Broß, 2009], parse^{pos} [Wu and Weld, 2010], KrakeN [Akbik and Löser, 2012], ClausIE [Del Corro and Gemulla, 2013], and [Gamallo et al., 2012] take advan-

tage of using dependency parsing in their extractor. These approaches are usually more expensive since they tradeoff efficiency for better accuracy and recall.

Inspired by the result of KnowItAll [Etzioni et al., 2004], other researches also considered extracting factual information from the Web. Matuszek et al. presented a method to enter new knowledge into Cyc by gathering and verifying facts from the World Wide Web [Matuszek et al., 2005]. In their approach, called *CycL*, they first select a query such as *(foundingAgent PalestineIslamicJihad ?WHO)* from their knowledge base, convert it to a search query, and search it on Google. The search engine results are then processed to extract a set of Ground Atomic Formula (GAF), e.g. *(foundingAgent PalestineIslamicJihad Terrorist-Nafi)*. After checking the consistency of the new set of GAFs with the knowledge already present in the knowledge base, a human volunteer reviews them for accuracy and adds them to the Cyc's KB. Similarly, [West et al., 2014a] developed an approach to complete the data in a knowledge base via search-based question answering. Their focus in this work is on learning the queries that should be issued to a search engine or a QA system in order to retrieve the most relevant documents.

Weakly supervised techniques are also used for large-scale IE techniques [Hoffmann et al., 2010; Mintz et al., 2009] where they used an existing ontology to generate a set of training data. This set of training data is later used to learn relation specific extractors. Although these techniques are able to learn relation-specific extractors on a large scale, it is not clear how easy it is to generalize these techniques since they are still restricted to use an input ontology. Preemptive IE [Shinyama and Sekine, 2006] and OnDemand IE [Sekine, 2006] are two examples of IE techniques that avoid relation specific extractors and do not require any input ontology. Instead, they rely on document and entity clustering. These dependencies make them too costly to be used in anytime applications.

7.3 Generic Relation Extraction

There has been considerable research in the community of information extraction focusing on supervised techniques to learn relation extractors[Califf and Mooney, 2003; Ciravegna, 2001; Soderland, 1999; Ramshaw et al., 2001]. The weakness of these techniques is

that they rely on assumptions, such as a large number of manually annotated examples, which make them unable to scale up to hundreds of relations. To deal with this weakness, others have developed unsupervised [Poon et al., 2010; Carlson et al., 2010b], weakly-supervised [Zhang, 2004; Lin et al., 2012], or bootstrap relation learning techniques [Brin, 1999; Agichtein and Gravano, 2000; Ravichandran and Hovy, 2002; Pantel and Pennacchiotti, 2006; Freedman et al., 2010] which starting from an initial set of seed examples, are able to generate a set of surface patterns or a set of rules that are later used to identify new relations.

Most of these works have been focused on precision. For example, [Ravichandran and Hovy, 2002] reported the results for the TREC question answering track where it is enough to only extract one instance of a relation. However, our focus in this thesis will be also on the recall. Extending some of these works, Freedman et al. [Freedman et al., 2011] showed that the recall of relation extraction techniques could be improved using bootstrap relation learning, handwritten patterns over predicate-argument structure, and coreference. The fact that they are dependent on a set of handwritten patterns limits the applicability of their techniques to be used for any generic relations.

Relation extractions for a small set of documents have been also studied by other researchers. For example Lin and Pantel developed an unsupervised method which uses syntactic patterns to discover inference rules from text [Lin and Pantel, 2001]. Stevenson presented a relation-extraction technique based on existing lexicons in the Wordnet [Stevenson, 2004]. Specia and Motta presented another relation extracting method that uses techniques such as part-of-speech tagger, named entity recognition, pattern-based classification, word sense disambiguation models, and resources such as input ontology to extract relations from the text [Specia and Motta, 2006].

Ontology-driver information extraction is another line of research in open information extraction that uses predefined ontologies. PANKOW [Cimiano et al., 2005; VÃlker, 2005] is an ontology-driven technique that queries Google using a set of Heast patterns, extracted from the input ontology, to annotate a set of named-entities in a document. Similar to this work, [Matuszek et al., 2005; McDowell and Cafarella, 2006] used ontologies such as Cyc combined with the search engines to identify semantic instances and relations. In contrast to these works that use predefined ontologies, Nigam developed a technique that

first automatically constructs an ontology by using Wikipedia's info boxes and then trains a CRF extractor using the extracted ontology [Nigam, 1999]. The fact that these techniques require a pre-defined ontology for information extraction limits their applicability and scalability to hundreds of relations.

A more general problem of relation extraction and open information extraction is Semantic Role Labeling (SRL). The main goal of SRL approaches is to identify arguments of verbs and the semantic relationship between them. Christensen et al. [Christensen et al., 2010] showed that SRL approaches can be used to increase precision and recall of open information extraction approaches. However, existing SLR approaches require a large amount of labeled data and are usually computationally expensive.

7.3.1 Relation Extraction Using Wikipedia

Wikipedia has been recently used as an important corpus to extract factual information. In this section, we briefly summarize different approaches that use Wikipedia, as a corpus for extracting semantic relations. These approaches is not directly relevant to what we are doing in this thesis since they are not directly applicable to free texts.

Ruiz-Casado et al. used Wikipedia to extract a set of relation types [Ruiz-casado et al., 2005; Ruiz-Casado et al., 2007]. Starting from two co-occurring semantically related WordNet nouns, documents in Wikipedia that contain both of these nouns are extracted. They then use the text that appears between these two noun phrases in a Wikipedia article to find the relations missing from WordNet. In later work, Ruiz-casado et al. showed that the accuracy of their technique could be improved by restricting their approach to Wikipedia pages that are written for specific categories [Ruiz-casado et al., 2006]. Along with this direction of research, Herbelot and Copestake used a dependency parser to identify subject, object, and their relationship in a sentence [Herbelot and Copestake, 2006]. Their experimental results showed that their technique was able to achieve high precision but at the cost of very low recall. Dependency parsers are also used by other researchers [Nguyen et al., 2007] to extract relation instances from Wikipedia. Suchanek et al. developed a pattern-based relation extraction technique which uses a context-free grammar for parsing a sentence and finding a set of patterns between two concepts [Suchanek

et al., 2006]. Machine learning techniques are then used to determine and generalize patterns that describe relations of interest. They showed that their approach outperforms other techniques, including a shallow pattern-matching resource TextToOnto [Cimiano and Völker, 2005].

Other researchers have used info boxes of Wikipedia for information extraction. For example, Wang et al. used positive seeds mentioned in the info boxes of Wikipedia in addition to using textual patterns that are extracted from the text [Wang et al., 2007b,c]. The same author further extended this work for the case that no negative examples are provided for training [Wang et al., 2007a]. Wu et al. used relation extraction in order to improve Wikipedia's info boxes [Wu and Weld, 2007, 2008]. To do this, they first use the content of info boxes, parse, and map them to a set of sentences in the text. Given these set of sentences, they train a classifier that later is used to classify new sentences. Hoffmann et al. also used a similar approach to create a set of training data by matching Wikipedia attribute values with corresponding sentences [Hoffmann et al., 2010]. They introduced a set of *dynamic lexicon features* which help to improve the accuracy of their system when learning from sparse data. They showed that their system, called LUCHS, is able to learn about 5025 relations.

7.4 Credibility of Information Sources

One of the main reasons for the success of the Web is the fact that it is open and decentralized which allows anyone to be able to directly contribute to the content of the Web. However, for a human or an agent to be able to use the information available on the Web and make decisions based on its information, they should measure the credibility of each source of information on the Web and to decide how it should be treated. Computers also have the challenge of making judgements given the available information on the Web because of the varying quality and truth that these diverse sources offer.

The wide need of measuring trust has made it a diverse topic [Golbeck, 2006; Artz and Gil, 2007] that has been studied in different areas of computer science. For example, *reputation-based systems* are based on the notion of transitive trust and are used to

measure the credibility of entities such as webpages [Page et al.], people [Levien et al., 1998], and peers in a network [Kamvar et al., 2003]. In *computational trust*, variations of probabilistic logic frameworks have been used to present a formal definition of trust and trust propagation, which can potentially be used in artificial agents to make trust-based decisions [Marsh, 1994; Manchala, 1998; Jøsang et al., 2006]. For *data integration*, different approaches are built to integrate the most complete and accurate records from a diverse set of sources where the data sources are extremely heterogeneous [Zhao et al., 2012; Dong and Srivastava, 2013; Li et al., 2014]. In *crowdsourcing*, various techniques have been developed to automatically validate the quality of crowd answers, by identifying the faulty workers [Venanzi et al.; Richardson and Domingos; Wang et al., 2013; Nguyen et al., 2015]. In *social networks*, Probabilistic Soft Logic (PSL) is used to model the trust in the social interactions [Huang et al., 2012, 2013]. The truthfulness of *deep web data* has been studied by [Li et al., 2013], which surprisingly shows a large amount of inconsistency between the data provided by different sources. In recommender systems [Abdul-Rahman and Hailes, 1997], P2P systems [Sherwood et al., 2006], and game theory [McCabe et al., 2003], the concept of trust is used to model the agent and player interactions [Jonker and Treur, 1999; Barber and Kim, 2001; Maes, 1994].

Since the scope of this thesis is to focus on measuring the trust of information that is extracted from an unstructured webpage, we restrict our definition of online trust to the trust that occurs when a user or a computer is communicating to an *informational website* on which we do not have any prior assumption on the type of the website or its topic. We also assume that to measure the trust score to a website, we do not use any explicit information/feedback that has been provided by the user about the trustworthiness/credibility of a website. This assumption is necessary since our focus in this thesis is to automatically measure trustworthiness of any unstructured website returned by a search engine.

Modeling the Trust

Since 1990, there has been extensive work conducted to understand what factors encourage users to trust a web site. The goal of this research is to determine the criteria that users consider to trust a website. The initial focus of this research was mostly on analyzing the criteria that affect trusting any e-commerce website [Falcone and Castelfranchi, 2001;

Riegelsberger et al., 2003; Salam et al., 2005; Van Slyke et al., 2004; Yang et al., 2005; Kim et al., 2011; Wang and Benbasat, 2008]. However, later researchers started studying what factors, in general, affect trust of an online website. For example, Corritore et al. showed that criteria such as the reputation of the author, ease of navigation, perception of credibility, and the design of a website affect the decision of users to trust a website [Corritore et al., 2003]. Similar to this work, Cyr [Cyr, 2008] studied how three factors [information design, navigation design, and visual design] affect a website's trustworthiness. In addition to this work, Josang et al. showed that trust is built over time [Jsang et al., 2005]. Fogg et al. generalized the concept of trust by studying what it means for a website to be *credible* [Fogg et al., 2001]. They defined credibility as measuring how likely a user would believe in the content of a website. They also showed that factors such as being cited (i.e., linked) by another trustworthy website, having a policy statement on the website, social recommendation (e.g., a friend recommends a website), and business interest (e.g., if it is a for-profit company) are the major criteria that affect the trustworthiness of a website.

Graph-based Trust Approaches

Unlike most of the approaches above which try to understand how users trust a given website, there has been other research focusing on using the link structure between different webpages to measure their credibility score. Initially, Page et al. introduced this idea by making the PageRank score biased to penalize spam websites [Page et al., 1999]. Bianchini et al. [Bianchini et al., 2005] and Langville et al. [Langville and Meyer, 2004] did further analyses on biased PageRank score. Gyongyi et al. were among the first who contributed a semi-automatic approach based on the link structure of the Web to separate reputable and good pages from spam webpages [Gyöngyi et al., 2004]. They first selected a small set of seed pages (i.e., webpages) that are evaluated by an expert. Starting from this set of reputable seed pages, they use the link structure of the Web to discover other pages that are likely to be good. The idea of using the link structure of the Web to predict spam websites has been further explored by other researchers [Castillo et al., 2007; Gan and Suel, 2007; Wu and Chellapilla, 2007]. In general, all of these approaches are subsets of Graph-based Semi-Supervised Learning (GSSL) algorithms, which are used to propagate information between different nodes of a graph. Given a few labeled nodes and a

graph where the edge weight represents the degree of similarity between the two connected nodes, GSSL algorithms classify the initially unlabeled nodes in the graph [Subramanya and Talukdar, 2014]. Since these algorithms can handle only a specific edge type (i.e., node similarity), they are not applicable to the credibility assessment graph that we have used in this thesis (Chapter 4) since a credibility assessment graph consists of multiple edge types with differing semantics.

Fact-finding Approaches

To overcome the limitations of GSSL algorithms (assuming that all the nodes have the same type), several fact-finding algorithms have been developed which consider a bipartite version of a graph similar to that of our credibility assessment graph, and propagate information between nodes in a non-symmetric way. Algorithms such as Sums [Kleinberg, 1999], TruthFinder [Yin et al., 2007], Generalized-Investment [Pasternack and Roth, 2011], Pooled-Investment [Pasternack and Roth, 2010], AccuVote [Dong et al., 2009], Average.Log [Pasternack and Roth, 2010], and 3-Estimates [Galland et al., 2010] have been developed that use different propagation update functions which specify how the information flows between the nodes in such a graph.

Each of these fact-finding algorithms suffers from at least one of the following main disadvantages. First, the score assigned to claims are *biased* toward favoring potentially (non-credible) sources that assert many evidences. This is primarly because these algorithms use either *average* or *sum* functions when aggregating credibility scores of websites containing relevant evidences. Second, the scores assigned to the nodes are not *interpretable*. For example, knowing that a source has a score of 10 without knowing the range and the possible distribution of the score is not informative, and therefore hard to interpret. Interpretability of the result is specially important when the output is used by either a human or by an application (which is one of the main motivations behind this thesis). Third, the convergence of these iterative algorithms is not guaranteed. Finally, and most importantly, incorporating additional prior-knowledge to guide how the credibility information should flow over the CA-like graph is not intuitive and easy. For example, the *majority voting* approach is based on the assumption (i.e., prior knowledge) that the truth score of each claim is computed by the weighted average of the sources connected to the claim, or

the Generalized Investment [Pasternack and Roth, 2010] approach assumes that sources "invest" their credibility uniformly in the claims that they make. These prior assumptions usually are incorporated through a set of cumbersome and usually non-intuitive update functions. Any change that needs to be made in these assumptions requires a change in the model.

Some of these limitations are partially addressed by other previous work. To *incorporate the prior knowledge*, Pasternack and Roth [Pasternack and Roth, 2010] introduced a pipelined approach which takes beliefs output by a fact-finder as an input, and "corrects" those beliefs based on some prior-knowledge defined by the user. Unlike our ClaimEval approach, [Pasternack and Roth, 2010] cannot incorporate additional prior-knowledge such as how credibility should propagate over the graph. The notion of prior-knowledge used in [Pasternack and Roth, 2010] is different than ours, since in our ClaimEval approach, prior knowledge can also refer to the way that the trust information propagates through the credibility graph. Additionally, and in contrast to ClaimEval, the semantics of belief scores in [Pasternack and Roth, 2010] is specific to the external fact-finder that is iteratively used to assign scores to the nodes in the graph, and convergence of the algorithm is also not guaranteed. To address *bias* and *interpretability* limitations of fact-finders, probabilistic fact-finding approaches are introduced that model the joint probability of the sources and the claims [Pasternack and Roth, 2013; Zhao et al., 2012; Wang et al., 2011]. In their models, the latent variable is the truth of the claim, and the probability of a claim being true is calculated using exact or approximate inference in a probabilistic graphical model. Although these approaches provide a transparent and interpretable model for the credibility analysis problem, the prior knowledge can be provided only in a node-centric manner. Incorporating more sophisticated types of prior-knowledge, such as adding different types of nodes or how information should propagate over the graph, requires non-trivial modeling changes. In contrast, ClaimEval offers a flexible framework which makes it possible to add such prior-knowledge using first-order logic rules and without requiring any changes in the model.

Moreover, these approaches do not provide an integrated system that asserts the truthfulness of claims by extracting evidences from the unstructured sources of information on the Web. OpenEval [Samadi et al., 2013] addresses this issue but doesn't take source cred-

ibility into account. Similarly, Defacto [Lehmann et al., 2012] only uses PageRank as an estimation for the credibility of source. Nakashole and Mitchell [Nakashole and Mitchell, 2014] introduced a language-aware approach based on NELL, called FactChecker, that makes use of linguistic feature to compute whether a source *objectively* states a fact, or if it is *opinionated*. In addition to the *objectivity* score, FactChecker also uses a *co-mention* score in order to assign similar credibility scores to fact candidates that are mentioned in similar sources. The final credibility score that is computed by the FactChecker is a combination of *objectively* and *co-mention* scores. FactChecker is incapable of incorporating prior credibility information into its model, and also focuses mostly on determining whether a source contains objective or subjective information. In contrast to these approaches, our ClaimEval approach identifies relevant sources, extracts evidences from them, estimates source credibility using the prior credibility knowledge, and uses those credibility scores for improved claim evaluation, all in an integrated system.

7.5 Searching for Objects

Exploring indoor environments has been studied as a part of the work on Simultaneous Localization and Mapping. Yamauchi described a search process that would incrementally explore new regions of the map [Yamauchi, 1997]. Our baseline search, which does not use background knowledge, is similar to this approach. Subsequent work focused on balancing the trade-off between localization quality and the need to explore new areas of the environment [Makarenko et al., 2002] [Stachniss et al., 2005].

Searching for objects has received considerable interest in the robotics community. Extensive research has focused on visual object search that does not leverage the Web [Sjöö et al., 2009; Aydemir et al., 2011; Velez et al., 2011; Joho et al., 2011]. Sjöö et al. presented a method for search and localization of objects by using an attention mechanism as a primary step in the recognition process[Sjöö et al., 2009]. Using a combination of view planning and visual search, the authors used existing computer vision algorithms to efficiently detect and localize different objects. Aydemir et al. built on this by using spatial relations to perform large-scale visual search for objects[Aydemir et al., 2011]. Joho et al. focused on the problem of finding an object with a mobile robot in an initially unknown,

structured environment[Joho et al., 2011]. While the primary focus is not on vision, they presented two methods for object search. The first was a reactive search technique based on objects in the robot's immediate vicinity. The second was a global, inference-based approach that uses the object arrangements of example environments. Finally, Velez et al. considered the task of autonomously navigating through the environment while mapping the location of objects [Velez et al., 2011] . The authors described an online any-time framework where vantage points provide the most informative view of an object given a noisy object detector. Unlike these approaches, our work uses help from humans to detect and manipulate objects.

7.6 Integration of Robots with the Web

Researchers have begun to consider how robots might be integrated with the Web. Meger et al. described an integrated robotic platform that uses web-based training data to train a visual object detector and then perform exploration, mapping, and active attention [Meger et al., 2008]. Most similar to this work is Kollar et al., who used the co-occurrences in the labels from the Flickr photo-sharing website as a prior over where objects are located in the physical environment [Kollar and Roy, 2009]. Posner et al. demonstrated a system that queries the Web to help read the visible text in the scene [Posner et al., 2010]. Tenorth et al. described how information on the World Wide Web, that is intended for human use, might be useful for robots [Tenorth et al., 2011].

7.7 Retrieving the Information from the Web for Planning

To the best of our knowledge, no work has been done on designing a general planner that automatically retrieves the information from the open Web while solving a planning problem. However, in the AI planning literature, there are several approaches developed for planning with incomplete-information which we briefly summarize in this section.

Most of the approaches for planning with incomplete information try to address the problem of incomplete knowledge by inserting sensing actions in the plan. These techniques usually generate a plan conditioned on the possible pieces of information that can be gathered by using sensing actions. [Etzioni et al., 1992] were among the first researchers who presented a planning language called UWL in order to handle incomplete knowledge. UWL distinguishes between actions that change the state of the world and actions that change the state of the agent's knowledge. Similarly, [Golden et al., 1996] developed XII planner which is an extension of UCPOP [Penberthy and Weld, 1992], and is able to generate sensing actions for information gathering during the execution. Similar to these works but using a different approach, [Pryor and Collins, 1996] developed a decision-based planning technique that uses explicit decision-steps that result in a set of subgoals to acquire knowledge. [Weld et al., 1998] extended the GraphPlan technique [Blum and Furst, 1997] to handle sensing actions by distinguishing between actions that sense the value of an unknown proposition from those that change its value.

In addition to these works, [chi Tuan et al., 2004] used belief states to deal with the incomplete information problem. Bonet and Geffner [Bonet and Geffner, 2000] presented a heuristic search approach to search in the belief state. The problem of planning with incomplete knowledge has been also addressed using logic programming. For example, [Son et al., 2004b,a] presented a conditional planner which is capable of generating both conditional plans and conformant plans in the presence of sensing actions. Similarly, [Rintanen, 1999] developed an approach to translate the conditional planning to quantified Boolean formulas.

Another line of research focuses on automatic Web Service Composition (WSC), where most of the information must be acquired from Web services. ENQUIRER [Kuter et al., 2004] is one of the examples of planners that can solve WSC problems. [Sohrabi et al., 2006] proposed a technique for WSC using preference-based learning technique [Bienvenu et al., 2006]. Similar to these approaches, other researchers have also developed techniques for WSC using approaches such as CSP [Kaldeli et al., 2009], HTN [Lin et al., 2008], and a combination of both HTN and CSP [Paik and Maruyama, 2007].

Our work differs from previous approaches in that it (i) assumes that the knowledge provided from the Web is uncertain and therefore should find a plan that has the highest

confidence value, (ii) iteratively adds knowledge to the planning problem as new instances are needed to solve the problem, (iii) does not rely on any specific Web services and is designed to query the Web for any type of knowledge that can be encoded as instances of predicates.

7.8 Budget-Sensitive Query Evaluation

Method	Is it test-time budget sensitive?	Can it handle multiple test-time budget without retraining?
Cascades [Raykar et al., 2010; Saberian and Vasconcelos, 2010; Chen et al., 2012]	No	No
CSTC [Xu et al., 2013]	No	No
AskMSR [Azari et al., 2004, 2012]	No	No
Kanani & McCallum [Kanani and McCallum, 2012]	No	No
GreedyMiser [Xu et al., 2012]	Yes	No
Karayev et al. [Karayev et al., 2013]	Yes	No
Weiss & Taskar [Weiss and Taskar, 2013b]	Yes	No
AskWorld [our approach]	Yes	Yes

Table 7.1: Comparison of characteristics of recently proposed feature-cost sensitive learning methods. The method presented in this thesis is the only one which is both test-time budget sensitive, and is flexible enough to handle multiple test-time budgets without retraining.

Our work in Chapter 3 leverages and extends some of the ideas explored by the previous work such as stacked classification [Wolpert, 1992b], learning classifier cascades [Saberian and Vasconcelos, 2010], active sensing and classification, learning under test-time budget, and dynamic feature selection. To explain the relationship between our AskWorld system and the previous work, we can think of KI as a classifier, with each

predicate-specific confidence from an expert becoming a feature in this classifier. Please note that overall, we can think of this as a classification problem with cost incurred during feature computation, and the cumulative cost upper bounded by a user specified limit. In this setting, the test-time budget may vary from one query to another, and hence the system should be flexible enough to handle such varying test-time budget constraints. In this section, we briefly describe related research on cost-sensitive feature acquisition and draw their connection to AskWorld.

Stacked classifiers aim at integrating classification scores from multiple baseline classifiers to produce the final classification [Wolpert, 1992b]. Even though this is very relevant for AskWorld, prior research in stacked classification has ignored any *budget considerations*, and hence they are not directly applicable to AskWorld. There is considerable previous work in the literature focusing on cascaded classifiers [Saberian and Vasconcelos, 2010]. The main idea behind these works is the fact that instead of acquiring all the features at once, one can acquire the inexpensive features at a very early stage and reject a set of test samples for which the classifier has a high confidence classification using only those inexpensive features. These algorithms iteratively acquire new features at each stage and reject some of the input test examples. This mechanism enables us not to acquire the most expensive features for the majority of test examples, and therefore reduces the effective average test cost. Early work on the cascaded classifiers focused on real-time object detection systems and assumed that all of the features have the same feature cost [Viola and Jones, 2001; Bourdev and Brandt, 2005; Zhang and Viola, 2007]. Raykar et al. [Raykar et al., 2010] extended the cascaded algorithms by jointly training different stages (i.e., classifiers) and reflecting the tradeoff between cost and accuracy during training. Saberian and Nuno extended Raykar et al.'s work by providing a precise mathematical model for cascade [Saberian and Vasconcelos, 2010]. Different from Raykar's work that pre-assigns features to cascade stages, Chen et al. proposed an algorithm that make the order of the feature extraction part of the training process [Chen et al., 2012]. However, none of these methods guarantee test classification within a pre-defined time budget and can not be directly applied to AskWorld.

Similar to the work on the cascaded classifiers, an approach that builds a rich family of classifiers during training which may vary in the computational cost (including cost

of computing feature values) is presented in [Gao and Koller, 2011]. During test time, the algorithm uses a myopic "value-of-information" computation to iteratively select base classifiers whose opinions should be integrated to classify the input test instance. Part of this model is learned during test time which introduces an additional cost during the test time. Unfortunately, this can be prohibitively expensive in the setting in which AskWorld operates.

Different from the above work, that stages feature extraction into linear cascades, Xu et al. [Xu et al., 2013] proposed a cost-sensitive tree of classifiers [Tan, 1993] which is a set of classifiers represented as a decision tree (each node is a classifier). During training, a decision tree is constructed to reduce the average test time of classification (including feature acquisition cost) while maximizing the accuracy. During the test, each node in the decision tree has a threshold which is used to send the input instance to different parts of the tree. A weight vector is also assigned to each node used for class prediction. The decision tree is constructed in a way that allocates higher feature budgets to infrequently traveled tree-paths. Unfortunately, this method is also not *test-time budget sensitive*.

To overcome this limitation, Xu et al. [Xu et al., 2012] proposed an algorithm that optimizes the cost-accuracy tradeoff by assuming that the test-time budget is known in advance during the training. Instead of the *"early exit"* assumption that lies behind most of the techniques on the classifier cascades, Xu et al. proposed an approach that builds a set of weak classifiers that uses the exact input test budget to classify a test instance (it is guaranteed that they provide an answer within the fixed test time budget). Similar to this work, Karayev et al. [Karayev et al., 2013] proposed a reinforcement learning approach that dynamically selects features which maximize the reward function. The reward is defined as the function of the information gain and the cost of each feature. Both of these techniques need to know the specific test-time budget in advance for training. Therefore, their applicability is limited when a different time budget is given as an input during the test. AskWorld aims to overcome this shortcoming by being flexible enough to handle varying test-time budgets without model retraining.

Similar to these works but applied in a question answering framework, Azari et al. [Azari et al., 2004, 2012] studied how to efficiently learn querying policies in a Web-based question answering system. Their system, *AskMSR*, first generates a set of Bayesian models

that can predict the accuracy of answers that are extracted for a given question. They then deploy a system which is able to dynamically control the number of queries that should be issued to the search engine. Their model is based on optimizing the net expected value of queries, computed as the difference of the expect value and cost. The *AskMSR* has many limitations: (i) all of the queries should have the same cost, (ii) only the optimal *number* of queries is computed in the model, without specifying the list of queries, and (iii) the learning algorithm optimizes the *expected cost* without considering the input budget constraint.

The prior research most relevant to ours is the method presented in [Weiss and Taskar, 2013b], where reinforcement learning techniques are used to learn a policy for feature selection under a specific test-time budgets. Even though a relaxation to handle varying test-time budget is presented in the same paper, it is not clear if *any* test-time budget can be complied with in the relaxed version of their model. It is worth mentioning that Kanani and McCallum [Kanani and McCallum, 2012] also presented a similar reinforcement learning approach to automatically correct the uncertain or missing content in a database. In their MDP formulation, the state of the database is represented by the states in the MDP where the actions determine how to extract information from the Web in order to fill missing slots in the database. Learning a policy without considering the budget constraint is the main drawback of this approach (the budget is used only during the plan execution). AskWorld aims to overcome these shortcomings by being able to handle in a principled manner varying test-time budget constraints without the need for retraining. It is worth mentioning that our work is also (loosely) related to some other work in task allocation in crowdsourcing systems [Karger et al., 2011; Chen et al., 2013a], information retrieval systems [Arnt et al., 2004], and timed zero-sum games [McMillen and Veloso, 2007].

7.9 How Our Work Fits

This thesis builds upon and deviates from several concepts in this array of related work:

- We study the problem of extracting information from the Web while a limited amount of time is given to our IE system.

- We assume that no ontology is provided as an input to our IE technique.
- We study how to make our IE techniques flexible enough to handle varying test-time budget constraints.
- We study the problem of integrating knowledge from different knowledge harvesting systems.
- We assume sources that are used for information extraction have different levels of credibility which affect the correctness of the information that they provide.
- We study how we can easily incorporate prior credibility knowledge into our IE system, and how to assign interpretable credibility scores to sources.
- We assume that our IE technique receives a feedback from an agent about the knowledge that it has provided. Our ClaimEval approach provides a flexible framework to incorporate this feedback to improve the accuracy of our IE technique.
- We study how our technique can be used in anytime applications. We measure the accuracy of our technique not only by testing it on a set of benchmarks, but also on how it performs on the applications.
- We study the problem of planning with incomplete information, and how to estimate and retrieve the missing relevant knowledge.
- We study how to learn background knowledge about the environment for a mobile robot by querying the Web.

Chapter 8

Conclusion

This dissertation seeks to answer the following question:

How can we build budget-sensitive, knowledge-on-demand models for jointly estimating the credibility of sources and the validity of queried claims using unstructured web information?

In this concluding chapter, we summarize the contributions that we have presented to answer this question, and to identify some future directions for research.

8.1 Thesis Contributions

- **Chapter 2: Anytime Approach for Web Information Query Evaluation**
  - A novel and fully automated technique, called OpenEval, that learns to evaluate the correctness of an input predicate instance using the Web. OpenEval requires a limited number of training data, in terms of seed examples and input ontology.
  - An approach for automatically enriching the input training data by extracting context words from the Web for the initial input seed examples. We present a new algorithm for the automatic detection of the main content block of any unstructured webpage.

- A learning approach that iteratively re-trains a set of classifiers in order to minimize the error of OpenEval. We formally define the *entropy value* of each trained classifier, and show that the iterative minimization of entropy values can help to improve the accuracy of the OpenEval approach. Our model also demonstrates that the accuracy of responses from OpenEval increases as more time is given for evaluation.
- An approach for automatically exploring information on the Web in order to disambiguate the input query instance (i.e., word sense disambiguation) during the test time.
- Experimental results showing that performance (F1 score) of the OpenEval approach is 2-4 times better than related techniques, in particular PMI and weakly-supervised classification approach [Zhang, 2004].
- The use of OpenEval in three different domains: robotics, drug discovery, and password-generation programs, and experimental results showing that OpenEval is able to correctly validate the correctness of new predicate instances for each of these domains.
- The use of OpenEval for the DrugBank, which is a comprehensive database containing information on drugs and drug targets. We show that by using OpenEval, errors in the database can be quickly identified and fixed. In particular, we show that OpenEval is able to detect the drug entries that were FDA-approved upon entry creation, but were never updated to reflect subsequent FDA withdrawls.
- The use of OpenEval in the Never-Ending Language Learning (NELL) system, and experimental results showing that OpenEval greatly improves the accuracy of NELL.

- **Chapter 3: Knowledge Integration and On-Demand Time-Budgeted Query Answering**
  - A novel Knowledge-on-Demand (KoD) service, called AskWorld, that aggregates opinions from multiple knowledge resources to return the most accurate response to a query.

- A budget-sensitive approach that is designed to provide a best-effort response within the time-budget specified by the user or the application. To the best of our knowledge, AskWorld is the first KoD system of its kind that is flexible enough to handle varying test-time budget constraints without model retraining. AskWorld achieves this by posing the problem as learning a policy in a Markov Decision Process (MDP).
- Experimental results showing that the accuracy of AskWorld improves when the system is allowed to issue queries corresponding to the non-query predicates in the Ontology, that is, predicates other than the one given in its input query.
- Extensive experiments on real-world datasets demonstrate the effectiveness of the AskWorld approach by outperforming the state-of-the-art approaches.

- **Chapter 4: Measuring Credibility of Sources and Extracting Reasons**

  - A novel and fully-integrated technique, called ClaimEval. Input to ClaimEval is the prior credibility assessment knowledge, and a set of claims, where the truth values of a few of them, is known. ClaimEval then evaluates the truth of a set of unlabeled claims by automatically crawling the relevant information from the Web (using the OpenEval approach), building the CA graph, calculating the credibility of sources, and incorporating the calculated credibility scores to validate the truth of the claims. ClaimEval is an extension of the OpenEval approach where it jointly estimates the credibility of sources and the correctness of claims. The accuracy of OpenEval significantly improves by incorporating the credibility score of information sources using the ClaimEval approach.
  - The use of Probabilistic Soft Logic (PSL) [Kimmig et al., 2012; Broecheler et al., 2010] in the ClaimEval approach, resulting in a *flexible* and *principled* framework for joint credibility estimation and claim evaluation. In contrast to previous approaches, ClaimEval has the following three properties: (1) ease of incorporation of prior knowledge, (2) guaranteed convergence, and (3) interpretable credibility scores. To the best of our knowledge, ClaimEval is the first

such integrated system of its kind.

  - The experimental results on real-world data demonstrating the effectiveness of ClaimEval compared to the state-of-the-art approaches.

- **Chapter 5: Using the Web to Interactively Learn to Find Objects**

  - The use of OpenEval to learn background knowledge about the environment by querying the Web, and in particular to evaluate the probability of predicate locationHasObject.
  - An approach for finding and delivering objects that dynamically instantiates a utility function using the results of a Web query, and which interactively learns about the physical environment by getting feedback from humans.
  - A demonstration of our system, enabling CoBot to find and fetch objects in a real-world environment.

- **Chapter 6: Iterative Query-Based Open World Planning**

  - An automated planning approach, called Open World Planner, that actively queries the Web to add needed knowledge until the planner solves the planning problem. Our open-world planning algorithm estimates the relevant knowledge to the state of a planning problem, retrieves it from the Web, and iteratively adds it to a classical planner.
  - Experimental results on a trip planning domain show that our technique is able to plan a trip for a variety of different problems, when no knowledge is given as input to the planner.

8.2 Future Research Directions

There are many promising directions for future research that are identified in this thesis. Some of them are discussed briefly below:

- **Sampling negative examples**: To train a classifier for predicate p, OpenEval needs to extract a set of positive and negative Context-Based Instances (CBI). Positive CBIs are obtained from the input seed examples of predicate p by searching the Web and processing the retrieved unstructured web pages. Negative CBIs are obtained by randomly sampling a set of CBIs for predicates that are mutually-exclusive to predicate p. There is a trade-off on the number of negative examples that should be used. If the negative examples heavily outnumber the positive examples, then the classifier will have a biased classification toward the negative class. On the other hand, by sampling a small portion of CBIs from all other mutually-exclusive predicates, we may lose some of the important negative examples that are needed to reduce the error of the classifier on an unseen test example. Drawing on different concepts that have been used in the machine learning community to sample a subset of negative examples [Robert et al., 1997; Kubat and Matwin, 1997; Akbani et al., 2004; Hong et al., 2007], it can be studied whether the accuracy of OpenEval will be improved by using a smarter strategy to select negative examples.

- **Detecting duplicated predicates**: To be able to scale up OpenEval, we also need to consider whether a new input predicate already exists in the current list of predicates that OpenEval is trained on. This is especially important since different names can be used to refer to the same concept. For example, predicates such as *located-at* or *place-of* may both refer to the same concept. Detecting similarity between predicates can be performed by considering ideas such as measuring the cosine similarity of CBIs that are extracted for different predicates.

- **Anytime aspect**: We have shown how OpenEval can effectively formulate several search queries to explore/exploit available information on the Web within a given time frame. We envision the anytime aspect of OpenEval to be further extended by using other factors such as the credibility of sources. Given a limited amount of time that is provided to OpenEval, we propose that it is more efficient to start processing the webpages which we trust the most. For example, given a limited time to measure the correctness of an input proposition, it may make more sense to start processing the wikipedia webpages compared to blogs since wikipedia is considered in general to be more trustworthy. One area for future research is to use our credibility assessment ap-

proach to decide the order of webpages that should be crawled and processed.

- **Incorporating feedback**: One of the main purposes of designing OpenEval is to provide knowledge for an agent. In Chapter 5, we showed a scenario when CoBot starts performing a task in an unknown environment where it does not have any information about the location types in the environment. We showed that CoBot is able to learn the location of objects by randomly moving in the environment, getting feedback from humans, and providing feedback to OpenEval. We further showed that our flexible ClaimEval approach is able to incorporate feedback in a principled way. Our approach can be extended by studying the type of feedback OpenEval can get from an agent and how to use this feedback. In particular, the following ideas can be explored: (i) Enabling OpenEval to self-recognize the predicates that need more training examples and to ask the agent to get feedback and provide new examples for such predicate (e.g., OpenEval can ask CoBot to go to the kitchen and check if coffee exists in the kitchen), (ii) feedback that is provided by humans might be noisy (e.g., error of speech-to-text component) or useful only in a specific environment (e.g., cooking pan is not expected to be found in kitchens in Gates Hillman Center, but is expected to be found in a kitchen in a restaurant). A flexible approach should assign a weight to each feedback that it receives, e.g., weights can be assigned based on the confidence of the agent on the feedback that it has provided.

- **Extending Open World Planner:** In Chapter 6, we developed a novel approach that automatically solves a planning problem by iteratively sending queries to OpenEval and acquiring knowledge that is relevant to the planning problem from the Web. Our experimental results show that our approach is able to plan a trip, when no knowledge is given as an input to the planner. In future work, a set of intensive experiments could be performed to show how our planner performs on different problems and domains. The Open World Planner approach can be further extended to include the knowledge that is relevant to both the initial and goal state.

- **Extending the credibility assessment approach:** In Chapter 4, we explained the details of our credibility assessment graph. Our approach can be further extended to estimate the trust score of a given website by considering a set of criterion that are defined

by users, e.g., researchers [Corritore et al., 2003; Cyr, 2008] have shown that the reputation of the author and design of the website can be used to measure the trust score of a website. Future work can study which factors should be used in OpenEval given the design principle of OpenEval which is to provide knowledge in anytime aspect. The value of the criteria that are used to measure the trust should be efficiently computable to be used in anytime applications.

- **Improving credibility assessment algorithm:** In Chapter 4, we explained the detail of our credibility assessment algorithm for jointly estimating credibility of sources and correctness of claims. ClaimEval uses Probabilistic Soft Logic (PSL), resulting in a flexible and principled framework which makes it easy to state and incorporate different forms of prior-knowledge. Although it is shown that the inference using probabilistic soft logic can be done in the polynomial time, the inference algorithm is still inefficient to be applied on large graphs. Future work should improve the efficiency of our credibility assessment algorithm and study properties of the algorithm such as convergence speed.

8.3 Concluding Remarks

In this thesis, we have introduced several information extraction models and approaches for learning to respond to the truth of facts using unstructured web information. We have focused on information extraction tasks initiated as queries from either automated agents or humans. We have evaluated our algorithms on a wide range of predicates chosen randomly from sources such as Freebase, NELL, and Wikipedia. We have also demonstrated that our information extraction techniques can be used to provide knowledge to anytime intelligent agents, in particular, for a find-deliver task in a real mobile robot (CoBot) and for a trip planner agent.

Appendix A

Domains used in the Open World Planner

This appendix contains the domains that we have used in our experiment for Open World Planner in Chapter 6. The experiments are run on three different types of trip planning domains with different levels of difficulties, which we present in the next. For each domain, ten different problem instances, with different start and destination locations, are used in our experiments.

A.1 Domain 1

In the first domain, our goal is to travel to a destination city and book a hotel in the city.

```
(define (domain trip-planning)
 (:requirements :strips)
 (:predicates
  (city ?x) (in-city ?x) (hotel ?x) (hotel-in-city ?x ?y) (booked-hotel ?x ?y))
 (:action book-hotel
  :parameters (?x ?y)
  :precondition   (and
```

```
      (in-city ?x)
      (hotel ?y)
      (hotel-in-city ?y ?x))
 :effect   (and (booked-hotel ?y ?x)))

(:action travel-city
 :parameters (?x ?y)
 :precondition   (and
      (in-city ?x)
      (city ?y))
 :effect   (and
      (in-city ?y)
      (not (in-city ?x))
     )))
```

A.2 Domain 2

In the second domain, we consider planning a trip to a destination city where the planner should also find a hotel and two attractions.

```
(define (domain trip-planning)
 (:requirements :strips)
 (:predicates
  (trip-planned ?x ?y)  (city ?x) (in-city ?x) (hotel ?x)
  (hotel-in-city ?x ?y) (booked-hotel-in-city ?x ?y)
  (museum-in-city ?x ?y) (museum ?x) (visited-attraction-in-city ?x ?y)
  (zoo-in-city ?x ?y) (zoo ?x) (city-has-beach ?x))

 (:action Visit˙Museum
 :parameters (?museum ?city)
 :precondition
  (and
   (museum-in-city ?museum ?city)
   (museum ?museum)
   (in-city ?city)
```

```
  (city ?city))
:effect
 (and (visited-attraction-in-city ?museum ?city)))

(:action Visit-Zoo
:parameters (?zoo ?city)
:precondition
 (and
  (in-city ?city)
  (city ?city)
  (zoo ?zoo)
  (zoo-in-city ?zoo ?city))
:effect
 (and (visited-attraction-in-city ?zoo ?city) ))

(:action Go-to-Beach
:parameters (?city)
:precondition
 (and
  (city ?city)
  (in-city ?city)
  (city-has-beach ?city))
:effect
 (and (visited-attraction-in-city Beach ?city)))

(:action Book-Hotel
 :parameters (?city ?hotel)
 :precondition
  (and
   (hotel ?hotel)
   (city ?city)
   (hotel-in-city ?hotel ?city))
 :effect
  (and (booked-hotel-in-city ?hotel ?city)))

(:action Travel-To-City
 :parameters (?x ?y)
 :precondition
```

```
  (and
   (in-city ?x)
   (city ?y))
 :effect
  (and
   (in-city ?y)
   (not (in-city ?x))
  ))

(:action Plan-A-Trip
 :parameters (?srcCity ?destCity ?attrA ?attrB ?hotel)
 :precondition
  (and
   (city ?srcCity)
   (city ?destCity)
   (in-city ?destCity)
   (visited-attraction-in-city ?attrA ?destCity)
   (visited-attraction-in-city ?attrB ?destCity)
   (booked-hotel-in-city ?hotel ?destCity)
   (not (= ?attrA ?attrB)))
 :effect
  (and (trip-planned ?srcCity ?destCity))))
```

A.3 Domain 3

In the third domain, our goal is to find a trip starting from an initial city to a destination city where another city is visited in between (the planner should find two attractions and one hotel in each city).

```
(define (domain trip-planning)
 (:requirements :strips)
 (:predicates
  (trip-planned ?x ?y) (city ?x) (in-city ?x) (hotel ?x)
  (hotel-in-city ?x ?y) (booked-hotel-in-city ?x ?y)
  (museum-in-city ?x ?y) (museum ?x) (visited-attraction-in-city ?x ?y)
```

```
 (visited-city ?x) (zoo-in-city ?x ?y) (zoo ?x) (city-has-beach ?x))

(:action Visit˙Museum
:parameters (?museum ?city)
:precondition
 (and
  (museum-in-city ?museum ?city)
  (museum ?museum)
  (in-city ?city))
:effect
 (and
  (visited-attraction-in-city ?museum ?city)))

(:action Visit-Zoo
:parameters (?zoo ?city)
:precondition
 (and
  (in-city ?city)
  (zoo ?zoo)
  (zoo-in-city ?zoo ?city))
:effect
 (and
  (visited-attraction-in-city ?zoo ?city)))

(:action Go-to-Beach
:parameters (?city)
:precondition
 (and
  (in-city ?city)
  (city-has-beach ?city))
:effect
 (and
  (visited-attraction-in-city Beach ?city)))

(:action Book-Hotel
 :parameters (?city ?hotel)
 :precondition
  (and
```

```
   (hotel ?hotel)
   (city ?city)
   (hotel-in-city ?hotel ?city))
 :effect
  (and
   (booked-hotel-in-city ?hotel ?city)))

(:action Travel-To-City
 :parameters (?x ?y ?z)
 :precondition
  (and
   (in-city ?x)
   (city ?y)
   (booked-hotel-in-city ?z ?y))
 :effect
  (and
   (in-city ?y)
   (not (in-city ?x))))

(:action Visit-Attractions-City
 :parameters (?city ?attrA ?attrB)
 :precondition
  (and
   (in-city ?city)
   (visited-attraction-in-city ?attrA ?city)
   (visited-attraction-in-city ?attrB ?city)
   (not (= ?attrA ?attrB)))
 :effect
  (and
   (visited-city ?city)))

(:action Plan-Trip
 :parameters (?srcCity ?destCity1 ?destCity2)
 :precondition
  (and
   (visited-city ?destCity1)
   (visited-city ?destCity2)
   (in-city ?destCity2)
```

```
    (not (= ?destCity1 ?destCity2))
    (not (= ?destCity1 ?srcCity))
    (not (= ?destCity2 ?srcCity)))
  :effect
   (and
    (trip-planned ?srcCity ?destCity2)))
)
```

Bibliography

Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms*, NSPW '97, pages 48–60, New York, NY, USA, 1997. ACM. ISBN 0-89791-986-6. 7.4

Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 85–94, New York, NY, USA, 2000. ACM. ISBN 1-58113-231-X. doi: 10.1145/336597.336644. URL http://doi.acm.org/10.1145/336597.336644. 7.3

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *In Proceedings of the 15th European Conference on Machine Learning (ECML*, pages 39–50, 2004. 8.2

Alan Akbik and Jürgen Broß. Wanderlust: Extracting Semantic Relations from Natural Language Text Using Dependency Grammar Patterns. July 2009. URL http://km.aifb.uni-karlsruhe.de/ws/semsearch09/semse2009˙7.pdf. 7.2

Alan Akbik and Alexander Löser. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 52–56, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2391200.2391210. 7.2

Aris Anagnostopoulos, Andrei Broder, and David Carmel. Sampling search-engine results. *World Wide Web*, 9(4):397–429, December 2006. 7.1

Andrew Arnt, Shlomo Zilberstein, James Allan, and Abdel-Illah Mouaddib. Dynamic composition of information retrieval techniques. *JIIS*, 23(1):67–97, 2004. 7.8

Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 5(2):58–71, June 2007. ISSN 1570-8268. 7.4

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, March 2003. ISSN 1532-4435. 2.2.3

Alper Aydemir, Kristoffer Sjöö, John Folkesson, Andrzej Pronobis, and Patric Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, May 2011. URL http://www.pronobis.pro/publications/aydemir2011icra. 7.5

David Azari, Eric Horvitz, Susan Dumais, and Eric Brill. Actions, answers, and uncertainty: A decision-making perspective on web-based question answering. *Inf. Process. Manage.*, 40(5):849–868, September 2004. ISSN 0306-4573. 7.8, 7.8

David Azari, Eric Horvitz, Susan T. Dumais, and Eric Brill. Web-based question answering: A decision-making perspective. *CoRR*, abs/1212.2453, 2012. 7.8, 7.8

Stephen Bach, Matthias Broecheler, Lise Getoor, and Dianne O leary. Scaling mpe inference for constrained continuous markov random fields with consensus optimization. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2654–2662. Curran Associates, Inc., 2012. 4.2.2

K. S. Barber and J. Kim. Belief revision process based on trust: Agents evaluating reputation of information sources. In *Trust in Cyber-societies, LNAI 2246*, pages 73–82. Springer, 2001. 7.4

Justin Betteridge, Alan Ritter, and Tom Mitchell. Assuming facts are expressed more than once, 2014. 2.3.7

Rajeev Bhattacharya, Timothy M. Devinney, and Madan M. Pillutla. A Formal Model of Trust Based on Outcomes. *The Academy of Management Review*, 23(3):459–472, 1998. ISSN 03637425. 4.1

Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Trans. Internet Technol.*, 5(1):92–128, February 2005. ISSN 1533-5399. 7.4

M. Bienvenu, C. Fritz, and S. McIlraith. Planning with qualitative temporal preferences. In *International Conference on Principles of Knowledge Representation and Reasoning (KR06)*, pages 134–144, Lake District, UK, June 2006. 7.7

Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'12, 2012. 5.2

Jeremiah Blocki, Manuel Blum, and Anupam Datta. Naturally rehearsing passwords. In *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 361–380. Springer, 2013. 2.1, 2.3.6

Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artif. Intell.*, 90(1-2):281–300, February 1997. ISSN 0004-3702. 6.2.2, 6.2.3, 7.7

Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, December 1997. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00063-5. URL http://dx.doi.org/10.1016/S0004-3702(97)00063-5. 2.2.2, 3.3.2

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08*, pages 1247–1250, 2008a. ISBN 978-1-60558-102-6. 2.1, 2.3.1, 3.4.1

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, 2008b. 3.1

Blai Bonet and Hctor Geffner. Planning with incomplete information as heuristic search in belief space. In *AIPS*, pages 52–61. AAAI Press, 2000. 7.7

L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243 vol. 2, 2005. 7.8

Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8. 3.4.3

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-intensive question answering. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*, pages 393–400, 2001. 7.1

Sergey Brin. Extracting patterns and relations from the world wide web. Technical Report 1999-65, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/421/. Previous number = SIDL-WP-1999-0119. 7.3

Matthias Broecheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. In *Uncertainty in Artificial Intelligence (UAI)*, 2010. 1.2, 4.1, 4.2.2, 4.2.2, 4.2.3, 8.1

Matt Gardner Jayant Krishnamurthy Ndapa Nakashole Mehdi Samadi Partha Talukdar Derry Wijaya Tom Mitche Bryan Kisiel, Justin Betteridge. Cmuml system for kbp 2013 slot filling. In *Proceedings of TAC 2013 Workshop.*, 2013. (document), 2.1, 2.3.7, 2.18

Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *J. Mach. Learn. Res.*, 4:177–210, December 2003. ISSN 1532-4435. doi: 10.1162/153244304322972685. URL http://dx.doi.org/10.1162/153244304322972685. 7.3

Jamie Callan and Eduard Hovy. Clueweb09 data set, 2009. http://boston.lti.cs.cmu.edu/Data/clueweb09/. 7.2

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010a. 2.3.7, 3.1, 3.4.1

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010b. 1, 7.2, 7.3

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 101–110, New York, NY, USA, 2010c. ACM. ISBN 978-1-60558-889-6. 7.2

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, 2010d. 1, 2.1, 2.3.7

Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 423–430, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277814. URL http://doi.acm.org/10.1145/1277741.1277814. 7.4

Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, Olivier Chapelle, and Dor Kedem. Classifier cascade for minimizing feature evaluation cost. In *AISTATS*, pages 218–226, 2012. 7.8, 7.8

Xi Chen, Qihang Lin, and Dengyong Zhou. In Sanjoy Dasgupta and David Mcallester, editors, *ICML*, volume 28, pages 64–72. JMLR Workshop and Conference Proceedings, May 2013a. 7.8

Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013b. 2.3.7

Le chi Tuan, Chitta Baral, Xin Zhang, and Tran Cao Son. Regression with respect to sensing actions and partial states. In *In Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAIÕ04*, pages 556–561. AAAI Press, 2004. 6.1, 7.7

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 52–60, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1866775.1866782. 7.3

Philipp Cimiano and Steffen Staab. Learning by googling. *SIGKDD Explor. Newsl.*, 6 (2):24–33, December 2004. ISSN 1931-0145. doi: 10.1145/1046456.1046460. URL http://doi.acm.org/10.1145/1046456.1046460. 7.1

Philipp Cimiano and Johanna Völker. Text2onto - a framework for ontology learning and data-driven change discovery. In Elisabeth Metais Andres Montoyo, Rafael Munoz, editor, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238, Alicante, Spain, Juni 2005. Springer. 7.3.1

Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme' the context: context-driven automatic semantic annotation with c-pankow. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 332–341, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. 7.3

Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, IJCAI'01, pages 1251–1256, San Francisco, CA, USA, 2001.

Morgan Kaufmann Publishers Inc. ISBN 1-55860-812-5, 978-1-558-60812-2. URL http://dl.acm.org/citation.cfm?id=1642194.1642261. 7.3

Robert T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583, 1989. URL http://ideas.repec.org/a/eee/intfor/v5y1989i4p559-583.html. 2.2.3

William W. Cohen. Minorthird, 2008. http://mloss.org/software/view/81/. 2.3.1

Cynthia L. Corritore, Beverly Kracher, and Susan Wiedenbeck. On-line trust: concepts, evolving themes, a model. *International Journal of Human-Computer Studies*, 58(6): 737 – 758, 2003. 7.4, 8.2

Valter Crescenzi and Giansalvatore Mecca. Automatic information extraction from large websites. *J. ACM*, 51(5):731–779, sep 2004. ISSN 0004-5411. 7.2

Dianne Cyr. Modeling web site design across cultures: Relationships to trust, satisfaction, and e-loyalty. *J. Manage. Inf. Syst.*, 24(4):47–72, April 2008. ISSN 0742-1222. 7.4, 8.2

Luciano Del Corro and Rainer Gemulla. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 355–366, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-2035-1. URL http://dl.acm.org/citation.cfm?id=2488388.2488420. 7.2

Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998. 2.2.3

Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Truth discovery and copying detection in a dynamic world. *Proc. VLDB Endow.*, 2(1):562–573, August 2009. ISSN 2150-8097. 7.4

X.L. Dong and D. Srivastava. Big data integration. In *ICDE*, 2013. 7.4

Doug Downey, Oren Etzioni, and Stephen Soderl. A probabilistic model of redundancy in information extraction. In *IJCAI*, pages 1034–1041, 2005. 7.1, 7.2

Doug Downey, Oren Etzioni, and Stephen Soderland. Analysis of a probabilistic model of redundancy in unsupervised information extraction. *Artif. Intell.*, 174(11):726–748, jul 2010. ISSN 0004-3702. 7.2

Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *In Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 115–125. Morgan Kaufmann, 1992. 6.1, 7.7

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. 1, 2.1, 2.3.3, 7.1, 7.2

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, dec 2008. ISSN 0001-0782. 1, 2.1, 7.2

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. Open information extraction: The second generation. In Toby Walsh, editor, *IJCAI*, pages 3–10. IJCAI/AAAI, 2011. 7.2

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011a. Association for Computational Linguistics. ISBN 978-1-937284-11-4. 6.2.6

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011b. Association for Computational Linguistics. ISBN 978-1-937284-11-4. 7.2

Rino Falcone and Cristiano Castelfranchi. Social trust: a cognitive approach. In Christiano Castelfranchi and Yao-Hua Tan, editors, *Trust and deception in virtual societies*, chapter Social trust: a cognitive approach, pages 55–90. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0-7923-6919-X. URL http://dl.acm.org/citation.cfm?id=379153.379157. 7.4

David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller. Watson: Beyond Jeopardy! *Artificial Intelligence*, 199-200:93–105, June 2013. ISSN 00043702. doi: 10.1016/j.artint.2012.06.009. URL http://dx.doi.org/10.1016/j.artint.2012.06.009. 1

B. J. Fogg, Jonathan Marshall, Othman Laraki, Alex Osipovich, Chris Varma, Nicholas Fang, Jyoti Paul, Akshay Rangnekar, John Shon, Preeti Swani, and Marissa Treinen. What makes web sites credible?: a report on a large quantitative study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 61–68, New York, NY, USA, 2001. ACM. ISBN 1-58113-327-8. 7.4

Marjorie Freedman, Edward Loper, Elizabeth Boschee, and Ralph Weischedel. Empirical studies in learning to read. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 61–69, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1866775.1866783. 7.3

Marjorie Freedman, Lance Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward, and Ralph Weischedel. Extreme extraction: machine reading in a week. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1437–1446, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145585. 7.3

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000. 3.4.3

Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 131–140, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. 7.4

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, ROBUS-UNSUP '12, pages 10–18, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2389961.2389963. 7.2

Diego Gambetta. *Trust: Making and Breaking Cooperative Relations*. B. Blackwell, 1990. 4.1

Qingqing Gan and Torsten Suel. Improving web spam classifiers using link structure. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb '07, pages 17–20, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-732-2. doi: 10.1145/1244408.1244412. URL http://doi.acm.org/10.1145/1244408.1244412. 7.4

Tianshi Gao and Daphne Koller. Active classification based on value of classifier. In *NIPS*, pages 1062–1070, 2011. 7.8

J. C. Gittins and D. M. Jones. A Dynamic Allocation Index for the Sequential Design of Experiments. In J. Gani, editor, *Progress in Statistics*, pages 241–266. North-Holland, Amsterdam, NL, 1974. 2.2.3

Jennifer Golbeck. Trust on the world wide web: a survey. *Found. Trends Web Sci.*, 1(2): 131–197, January 2006. 7.4

Keith Golden, Oren Etzioni, and Daniel Weld. Planning with execution and incomplete information. Technical report, Department of Computer Science, University of Washington, 1996. 6.1, 7.7

Google. Google knowlede graph - http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html, 2012. 1

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, March 2002. ISSN 0885-6125. 3.3.2

Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 576–587. VLDB Endowment, 2004. ISBN 0-12-088469-0. 7.4

Sanda Harabagiu and Andrew Hickl. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 905–912, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. 1, 2.1

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. 7.2

Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006a. 6.2, 6.2.7

Malte Helmert. The fast downward planning system. *J. Artif. Int. Res.*, 26(1):191–246, July 2006b. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=1622559.1622565. 1.1.4

Aurelie Herbelot and Ann Copestake. Acquiring ontological relationships from wikipedia using rmrs. In *In ISWC 2006 Workshop on Web Content*, 2006. 7.3.1

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 286–295, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 7.2, 7.3.1

Xia Hong, Sheng Chen, and C. J. Harris. A kernel-based two-class classifier for imbalanced data sets. *Trans. Neur. Netw.*, 18(1):28–41, January 2007. ISSN 1045-9227. doi: 10.1109/TNN.2006.882812. URL http://dx.doi.org/10.1109/TNN.2006.882812. 8.2

Bert Huang, Stephen H. Bach, Eric Norris, Jay Pujara, and Lise Getoor. Social group modeling with probabilistic soft logic. In *NIPS Workshop on Social Network and Social Media Analysis: Methods, Models, and Applications*, 2012. 7.4

Bert Huang, Angelika Kimmig, Lise Getoor, and Jennifer Golbeck. A flexible framework for probabilistic models of social trust. In *International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction (SBP)*, 2013. 7.4

Dominik Joho, Martin Senk, and Wolfram Burgard. Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems*, 59(5):319–328, May 2011. ISSN 0921-8890. doi: 10.1016/j.robot.2011.02.012. URL http://ais.informatik.uni-freiburg.de/publications/papers/joho11ras.pdf. 7.5

Catholijn M. Jonker and Jan Treur. Formal analysis of models for the dynamics of trust based on experiences. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: MultiAgent System Engineering*, MAAMAW '99, pages 221–231, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66281-2. 7.4

Audun Jøsang, Stephen Marsh, and Simon Pope. Exploring different types of trust propagation. In *Proceedings of the 4th International Conference on Trust Management*, 2006. 7.4

Audun Jsang, Claudia Keser, and Theo Dimitrakos. Can we manage trust? In *Proceedings of the Third International Conference on Trust Management (iTrust), Versailes*, pages 93–107. Springer-Verlag, 2005. 7.4

Eirini Kaldeli, Alexander Lazovik, and Marco Aiello. Extended goals for composing services. In *ICAPS*, 2009. 7.7

Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, 2003. 7.4

Pallika H. Kanani and Andrew K. McCallum. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of WSDM*, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. 7.8, 7.8

Sergey Karayev, Mario Fritz, and Trevor Darrell. Dynamic feature selection for classification on a budget. In *ICML Workshop on Prediction with Sequential Models*, 2013. 3.1.1, 7.8, 7.8

David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *CoRR*, abs/1110.3564, 2011. 7.8

Myung-Ja Kim, Namho Chung, and Choong-Ki Lee. The effect of perceived trust on electronic commerce: Shopping online for tourism products and services in south korea. *Tourism Management*, 32(2):256 – 265, 2011. 7.4

Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*, 2012. 1.2, 4.2.2, 4.2.2, 8.1

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5): 604–632, September 1999. ISSN 0004-5411. 4.3.2, 2, 7.4

C. Knox, V. Law, T. Jewison, P. Liu, S. Ly, A. Frolkis, A. Pon, K. Banco, C. Mak, V. Neveu, Y. Djoumbou, R. Eisner, A. C. Guo, and D. S. Wishart. DrugBank 3.0: a comprehensive resource for 'omics' research on drugs. *Nucleic Acids Res.*, 39(Database issue):D1035–1041, Jan 2011. (document), 2.3.6, 2.14

Thomas Kollar and Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. In *Proceedings of the IEEE international conference on Robotics and Automation*, ICRA'09, pages 4116–4121, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2788-8. 2.1, 2.3.6, 5.1, 5.3.1, 7.6

Thomas Kollar, Mehdi Samadi, and Manuela Veloso. Enabling robots to find and fetch objects by querying the web. In *Proceedings of the 11th International Conference on*

Autonomous Agents and Multiagent Systems - Volume 3, AAMAS '12, pages 1217–1218, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9817381-3-3, 978-0-9817381-3-0. URL http://dl.acm.org/citation.cfm?id=2343896.2343929. 2.3.6

Jayant Krishnamurthy and Tom M. Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 754–765, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390948.2391030. 2.3.7

Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997. 8.2

Miroslav Kubat, Robert Holte, and Stan Matwin. Learning when negative examples abound. In Maarten van Someren and Gerhard Widmer, editors, *Machine Learning: ECML-97*, volume 1224 of *Lecture Notes in Computer Science*, pages 146–153. Springer Berlin / Heidelberg, 1997. ISBN 978-3-540-62858-3. 2.2.2

Ugur Kuter, Evren Sirin, Dana Nau, Bijan Parsia, and James Hendler. Information gathering during planning for web service composition. In *Journal of Web Semantics*, pages 335–349, 2004. 6.1, 7.7

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813. 2.3.7

Michail G. Lagoudakis, Ronald Parr, and L. Bartlett. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:2003, 2003. 3.3.2

Amy N. Langville and Carl D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1: 2004, 2004. 7.4

Richard C. T. Lee. Fuzzy logic and the resolution principle. *J. ACM*, 19(1):109–119, January 1972. 6.2.5

Jens Lehmann, Daniel Gerber, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. Defacto - deep fact validation. In *ISWC*, 2012. 7.4

Raph Levien, Alex Aiken, Raph Levien, and Alexander Aiken. Attack resistant trust metrics for public key certification. In *In 7th USENIX Security Symposium*, 1998. 7.4

Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, 2014. 7.4

Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: is the problem solved? In *VLDB*, 2013. 7.4

Chenghua Lin, Yulan He, Richard Everson, and Stefan Ruger. Weakly supervised joint sentiment-topic detection from text. *IEEE Transactions on Knowledge and Data Engineering*, 24:1134–1145, 2012. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.48. 7.3

Dekang Lin and Patrick Pantel. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 323–328, New York, NY, USA, 2001. ACM. ISBN 1-58113-391-X. doi: 10.1145/502512.502559. URL http://doi.acm.org/10.1145/502512.502559. 7.3

Naiwen Lin, Ugur Kuter, and Evren Sirin. Web service composition with user preferences. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 629–643, 2008. 7.7

Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from web documents. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 588–593, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775134. URL http://doi.acm.org/10.1145/775047.775134. 2.2.1, 2.2.1

Pattie Maes. Agents that reduce work and information overload. *Commun. ACM*, 37(7): 30–40, July 1994. ISSN 0001-0782. 7.4

Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 425–432, 2002. 1, 2.1, 7.1

A.A. Makarenko, S.B. Williams, F. Bourgault, and H.F. Durrant-Whyte. An experiment in integrated exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 534–539. IEEE, 2002. 7.5

Daniel W. Manchala. Trust metrics, models and protocols for electronic commerce transactions. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, 1998. 7.4

Stephen Paul Marsh. Formalising trust as a computational concept. Technical report, Department of Computing Science and Mathematics, University of Stirling, 1994. 7.4

Cynthia Matuszek, Michael Witbrock, Robert C. Kahlert, John Cabral, Dave Schneider, Purvesh Shah, and Doug Lenat. Searching for common sense: populating cyc™ from the web. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, AAAI'05, pages 1430–1435. AAAI Press, 2005. 7.2, 7.3

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. In *EMNLP-CoNLL*, pages 523–534. ACL, 2012. ISBN 978-1-937284-43-5. 7.2

Kevin A. McCabe, Mary L. Rigdon, and Vernon L. Smith. Positive reciprocity and intentions in trust games. *Journal of Economic Behavior & Organization*, 52(2):267–275, October 2003. 7.4

Luke K. McDowell and Michael Cafarella. Ontology-driven information extraction with ontosyphon. In *Proceedings of the 5th international conference on The Semantic Web*, ISWC'06, pages 428–444, Berlin, Heidelberg, 2006. Springer-Verlag. 7.3

Colin McMillen and Manuela M. Veloso. Thresholded rewards: Acting optimally in timed, zero-sum games. In *AAAI*, pages 1250–1255. AAAI Press, 2007. 7.8

David Meger, Per-Erik Forssén, Kevin Lai, Scott Helmer, Sancho McCann, Tristram Southey, Matthew Baumann, James J. Little, and David G. Lowe. Curious george: an attentive semantic robot. *Robotics and Autonomous Systems*, 56:503–511, June 2008. ISSN 0921-8890. 7.6

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-46-6. URL http://dl.acm.org/citation.cfm?id=1690219.1690287. 2.3.7, 7.2

T. Mitchell, W. Cohen, E. Hruscha, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohammad, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *AAAI*, 2015a. URL http://www.cs.cmu.edu/~wcohen/pubs.html. (document), 2.1, 2.3.7, 2.15

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of AAAI*, 2015b. 2.1

Dunja Mladenić, Janez Brank, Marko Grobelnik, and Natasa Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 234–241, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. doi: 10.1145/1008992.1009034. URL http://doi.acm.org/10.1145/1008992.1009034. 2.2.2

Ndapandula Nakashole and Tom M. Mitchell. Language-aware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1019, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-1095. 7.4

Preslav Nakov and Marti Hearst. A study of using search engine page hits as a proxy for n-gram frequencies. In *In Proceedings of the RANLPâ05*, 2005a. 7.1

Preslav Nakov and Marti Hearst. Search engine statistics beyond the n-gram: application to noun compound bracketing. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 17–24, Stroudsburg, PA, USA, 2005b. Association for Computational Linguistics. 7.1

Preslav Nakov and Marti Hearst. Using the web as an implicit training set: application to structural ambiguity resolution. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 835–842, Stroudsburg, PA, USA, 2005c. Association for Computational Linguistics. 7.1

Dat P. T. Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. Relation extraction from wikipedia using subtree mining. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pages 1414–1420. AAAI Press, 2007. ISBN 978-1-57735-323-2. 7.3.1

Quoc Viet Hung Nguyen, Chi Thang Duong, Matthias Weidlich, and Karl Aberer. Minimizing Efforts in Validating Crowd Answers. In *SIGMOD*, 2015. 7.4

Kamal Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999. 7.3

Charles Ogden. Basic english. a general introduction with rules and grammar. London, 1944. Paul Treber Co. ISBN 1-58113-702-8. doi: http://doi.acm.org/10.1145/985692.985733. URL http://doi.acm.org/10.1145/985692.985733. 2.3.6

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. 7.4

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120. 7.4

I. Paik and D. Maruyama. Automatic web services composition using combining htn and csp. In *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*, pages 206 –211, oct. 2007. 7.7

Patrick Pantel and Marco Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 113–120, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220190. URL http://dx.doi.org/10.3115/1220175.1220190. 7.3

Jeff Pasternack and Dan Roth. Knowing what to believe (when you already know something). In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 877–885, Beijing, China, August 2010. Coling 2010 Organizing Committee. 4.3.2, 7.4

Jeff Pasternack and Dan Roth. Making better informed trust decisions with generalized fact-finding. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2324–2329. AAAI Press, 2011. ISBN 978-1-57735-515-1. 4.3.2, 2, 3, 5, 6, 4.3.4, 7.4

Jeff Pasternack and Dan Roth. Latent credibility analysis. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1009–1020, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-2035-1. 7.4

J. Scott Penberthy and Daniel S. Weld. Ucpop: A sound, complete, partial order planner for adl. In *KR-92*, pages 103–114. Morgan Kaufmann, 1992. 7.7

Vittorio Perera, Robin Soetens, Thomas Kollar, Mehdi Samadi, Yichao Sun, Daniele Nardi, Rene van de Molengraft, and Manuela Veloso. Learning task knowledge from dialog and web access. *Submitted to Journal of Robotics*, 2015. 5.3, 5.3.3

Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Mausam, Alan Ritter, Stefan Schoenmackers, Stephen Soderland, Dan Weld, Fei Wu, and Congle Zhang. Machine reading at the university of washington. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 87–95, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 7.2, 7.3

Ingmar Posner, Peter Corke, and Paul Newman. Using text-spotting to query the world. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010. 7.6

Louise Pryor and Gregg Collins. Planning for contingencies: a decision-based approach. *J. Artif. Int. Res.*, 4(1):287–339, May 1996. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=1622737.1622749. 7.7

Lance Ramshaw, Elizabeth Boschee, Sergey Bratus, Scott Miller, Rebecca Stone, Ralph Weischedel, and Alex Zamanian. Experiments in multi-modal automatic content extraction. In *Proceedings of the first international conference on Human language technology research*, HLT '01, pages 1–5, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. doi: 10.3115/1072133.1072176. URL http://dx.doi.org/10.3115/1072133.1072176. 7.3

Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 41–47, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073092. URL http://dx.doi.org/10.3115/1073083.1073092. 7.3

Vikas C. Raykar, Balaji Krishnapuram, and Shipeng Yu. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In *KDD*, pages 853–860. ACM, 2010. 7.8, 7.8

Matthew Richardson and Pedro Domingos. Building large knowledge bases by mass collaboration. In *Proceedings of the 2nd International Conference on Knowledge Capture*. 7.4

Jens Riegelsberger, M. Angela Sasse, and John D. McCarthy. Shiny happy people building trust?: photos on e-commerce websites and consumer trust. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 121–128, New York, NY, USA, 2003. ACM. ISBN 1-58113-630-7. 7.4

Jussi Rintanen. Constructing conditional plans by a theorem-prover. *J. Artif. Int. Res.*, 10 (1):323–352, May 1999. ISSN 1076-9757. 7.7

Miroslav Kubat Robert, Robert Holte, and Stan Matwin. Learning when negative examples abound. In *In ECML-97, Lecture Notes in Artificial Intelligence*, pages 146–153. Springer Verlag, 1997. 8.2

Stephanie Rosenthal, Joydeep Biswas, and Manuela Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, 2010. 5.2

Ronald Rousseau. Daily time series of common single word searches in altavista and northernlight. *Cybermetrics*, 2/3(1), 1999. 7.1

Maria Ruiz-casado, Enrique Alfonseca, and Pablo Castells. Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In *In NLDB*, pages 67–79. Springer Verlag, 2005. 7.3.1

Maria Ruiz-casado, Enrique Alfonseca, and Pablo Castells. From wikipedia to semantic relationships: a semi-automated annotation approach. In *1st Workshop on Semantic Wikis: From Wiki to Semantics, at the 3rd European Semantic Web Conference (ESWC 2006). Budva*, 2006. 7.3.1

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. Automatising the learning of lexical patterns: An application to the enrichment of wordnet by extracting semantic relationships from wikipedia. *Data Knowl. Eng.*, 61(3):484–499, June 2007. ISSN 0169-023X. 7.3.1

Mohammad J. Saberian and Nuno Vasconcelos. Boosting classifier cascades. In *NIPS*, pages 2047–2055, 2010. 3.1.1, 7.8, 7.8

A.F. Salam, Lakshmi Iyer, Prashant Palvia, and Rahul Singh. Trust in e-commerce. *Commun. ACM*, 48(2):72–77, February 2005. ISSN 0001-0782. 7.4

Mehdi Samadi, Thomas Kollar, and Manuela M. Veloso. Using the web to interactively learn to find objects. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012. 1, 2.3.6, 3.1

Mehdi Samadi, Manuela Veloso, and Manuel Blum. Openeval: Web information query evaluation. In *Proceedings of the Twenty-Seventh National Conference on Artificial Intelligence*, AAAI13, 2013. 7.4

Satoshi Sekine. On-demand information extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 731–738, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1273073.1273167. 7.2

Rob Sherwood, Seungjoon Lee, and Bobby Bhattacharjee. Cooperative peer groups in nice. *Comput. Netw.*, 50(4):523–544, March 2006. ISSN 1389-1286. 7.4

Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 304–311, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220835.1220874. URL http://dx.doi.org/10.3115/1220835.1220874. 7.2

Kristoffer Sjöö, Dorian Gálvez López, Chandana Paul, Patric Jensfelt, and Danica Kragic. Object search and localization for an indoor mobile robot. *Journal of Computing and Information Technology*, 17(1):67–80, 2009. doi:10.2498/cit.1001182. 7.5

Stephen Soderl, Oren Etzioni, Tal Shaked, and Daniel S. Weld. The use of web-based statistics to validate information extraction. In *AAAI Workshop on Adaptive Text Extraction and Mining*, July 2004. 7.1

Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Mach. Learn.*, 34(1-3):233–272, February 1999. ISSN 0885-6125. doi: 10.1023/A:1007562322031. URL http://dx.doi.org/10.1023/A:1007562322031. 7.3

Shirin Sohrabi, Nataliya Prokoshyna, and Sheila A. Mcilraith. Web service composition via generic procedures and customizing user preferences. In *Int. Semantic Web Conf., ISWC2006*, pages 597–611, 2006. 6.1, 7.7

Tran C. Son, Huy, and Chitta Baral. Planning with Sensing Actions and Incomplete Information using Logic Programming. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, volume 2923 of *Lecture Notes in Computer Science*, pages 261–274, Fort Lauderdale, FL, USA, 2004a. Springer. 7.7

Tran Cao Son, Phan Huy Tu, and Chitta Baral. Planning with sensing actions and incomplete information using logic programming. In Vladimir Lifschitz and Ilkka NiemelŁ, editors, *Logic Programming and Nonmonotonic Reasoning, 7th International Conferenc*, volume 2923 of *Lecture Notes in Computer Science*, pages 261–274. Springer, 2004b. 7.7

Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 203–211, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: 10.1145/988672.988700. URL http://doi.acm.org/10.1145/988672.988700. 2.2.1

Lucia Specia and Enrico Motta. M.: A hybrid approach for extracting semantic relations from texts. In *In. Proceedings of the 2 nd Workshop on Ontology Learning and Population*, pages 57–64, 2006. 7.3

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of robotics: science and systems (RSS)*, pages 65–72, 2005. 7.5

M. Stevenson. An Unsupervised WordNet-based Algorithm for Relation Extraction. In *Proceedings of the "Beyond Named Entity" workshop at the Fourth International Conference on Language Resources and Evalutaion (LREC'04)*, 2004. 7.3

Amarnag Subramanya and Partha Pratim Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014. 7.4

Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the*

12th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '06, pages 712–717, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. 7.3.1

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of WWW*, 2007. 3.1

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, pages 216–224. Morgan Kaufmann, 1990. 2.2.3

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. 3.3.2, 3.3.2

Kartik Talamadupula, Gordon Briggs, Matthias Scheutz, and Subbarao Kambhampati. Architectural mechanisms for handling human instructions in open-world mixed-initiative team tasks. In *Advances in Cognitive Systems (ACS)*, 2013. 6.1

Ming Tan. Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning*, 13(1):7–33, 1993. 7.8

M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *Robotics & Automation Magazine, IEEE*, 18(2):58–68, 2011. 7.6

Peter Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl, 2001. 2.1, 1, 2.3.3, 7.1

Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424, Morristown, NJ, USA, 2002. Association for Computational Linguistics. 7.1

Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21:315–346, October 2003. ISSN 1046-8188. 7.1

Ahmet Uyar. Investigation of the accuracy of search engine hit counts. *J. Inf. Sci.*, 35: 469–480, August 2009. ISSN 0165-5515. 7.1

Craig Van Slyke, France Belanger, and Christie L. Comunale. Factors influencing the adoption of web-based shopping: the impact of trust. *SIGMIS Database*, 35(2):32–49, June 2004. 7.4

Johanna VÃlker. Towards large-scale, open-domain and ontology-based named entity classification. In *In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLPâ05*, pages 166–172. INCOMA Ltd, 2005. 7.3

J. Velez, G. Hemann, A. Huang, I. Posner, and N. Roy. Planning to perceive: Exploiting mobility for robust object detection. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Freiburg, Germany, 2011. 7.5

Manuela M. Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Thomas Kollar, etin Merili, Mehdi Samadi, Susana Brando, and Rodrigo Ventura. Cobots: Collaborative robots servicing multi-floor buildings. In *IROS*, pages 5446–5447. IEEE, 2012. ISBN 978-1-4673-1737-5. 2.1

Matteo Venanzi, Alex Rogers, and Nicholas R. Jennings. Trust-based fusion of untrustworthy information in crowdsourcing applications. In *AAMAS 2013*. 7.4

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. 7.8

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: http://doi.acm.org/10.1145/985692.985733. URL http://doi.acm.org/10.1145/985692.985733. 2.3.6, 2.3.6, 5.3.1

Dong Wang, Tarek Abdelzaher, Hossein Ahmadi, Jeff Pasternack, Dan Roth, Manish Gupta, Jiawei Han, Omid Fatemieh, Hieu Le, and Charu C. Aggarwal. On Bayesian interpretation of fact-finding in information networks. In *Information Fusion*, 2011. 7.4

Dong Wang, T. Abdelzaher, L. Kaplan, and C.C. Aggarwal. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. In *ICDCS*, 2013. 7.4

Gang Wang, Yong Yu, and Haiping Zhu. Pore: positive-only relation extraction from wikipedia text. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 580–594, Berlin, Heidelberg, 2007a. Springer-Verlag. 7.3.1

Gang Wang, Huajie Zhang, Haofen Wang, and Yong Yu. Enhancing relation extraction by eliciting selectional constraint features from wikipedia. In Zoubida Kedad, Nadira Lammari, Elisabeth MÃ©tais, Farid Meziane, and Yacine Rezgui, editors, *Natural Language Processing and Information Systems*, volume 4592 of *Lecture Notes in Computer Science*, pages 329–340. Springer Berlin / Heidelberg, 2007b. 7.3.1

Pu Wang, Jian Hu, Hua-Jun Zeng, Lijun Chen, and Zheng Chen. Improving text classification by using encyclopedia knowledge. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 332–341, Washington, DC, USA, 2007c. IEEE Computer Society. 7.3.1

Richard C. Wang and William W. Cohen. Automatic Set Instance Extraction using the Web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 441–449. Association for Computational Linguistics, August 2009. 2.3.7

Weiquan Wang and Izak Benbasat. Attributions of trust in decision support technologies: A study of recommendation agents for e-commerce. *J. Manage. Inf. Syst.*, 24(4):249–273, April 2008. ISSN 0742-1222. 7.4

David J Weiss and Ben Taskar. Learning adaptive value of information for structured prediction. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 953–961, 2013a. 3.3.1

David J Weiss and Ben Taskar. Learning adaptive value of information for structured prediction. In *Advances in Neural Information Processing Systems*, 2013b. 7.8, 7.8

Daniel S. Weld, Corin R. Anderson, and David E. Smith. Extending graphplan to handle uncertainty and sensing actions. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 897–904, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL http://dl.acm.org/citation.cfm?id=295240.295920. 6.1, 7.7

Tim Weninger, William H. Hsu, and Jiawei Han. Cetr: Content extraction via tag ratios. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 971–980, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772789. URL http://doi.acm.org/10.1145/1772690.1772789. 2.2.1

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *WWW*, 2014a. URL http://www.cs.ubc.ca/~murphyk/Papers/www14.pdf. 7.2

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *WWW*, 2014b. 3.1

David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992a. 2.3.7

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992b. 7.8

Baoning Wu and Kumar Chellapilla. Extracting link spam using biased random walks from spam seed sets. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb '07, pages 37–44, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-732-2. doi: 10.1145/1244408.1244416. URL http://doi.acm.org/10.1145/1244408.1244416. 7.4

Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 41–50, New York, NY, USA, 2007. ACM. 7.3.1

Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 635–644, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. 7.3.1

Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858681.1858694. 7.2

Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. The greedy miser: Learning under test-time budgets. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1175–1182, New York, NY, USA, July 2012. ACM. ISBN 978-1-4503-1285-1. URL http://icml.cc/2012/papers/592.pdf. 3.1.1, 3.4.3, 3.4.3, 7.8, 7.8

Zhixiang Xu, Matt Kusner, Minmin Chen, and Kilian Q. Weinberger. Cost-sensitive tree of classifiers. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 133–141. JMLR Workshop and Conference Proceedings, 2013. 3.1.1, 7.8, 7.8

B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997. 7.5

Yun Yang, Yong Hu, and Juhua Chen. A web trust-inducing model for e-commerce and empirical research. In *Proceedings of the 7th international conference on Electronic commerce*, ICEC '05, pages 188–194, New York, NY, USA, 2005. ACM. ISBN 1-59593-112-0. 7.4

Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 296–305, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956785. URL http://doi.acm.org/10.1145/956750.956785. 2.2.1, 2.2.1

Xiaoxin Yin, Jiawei Han, and Philip S. Yu. Truth discovery with multiple conflicting information providers on the web. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 1048–1052, New York, NY, USA, 2007. ACM. 4.3.2, 4, 7.4

Cha Zhang and Paul A. Viola. Multiple-instance pruning for learning efficient cascade detectors. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*, 2007. 7.8

Zhu Zhang. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 581–588, New York, NY, USA, 2004. ACM. ISBN 1-58113-874-1. 2.1, 1, 2.3.3, 2.3.4, 2.4, 7.3, 8.1

Bo Zhao, Benjamin I. P. Rubinstein, Jim Gemmell, and Jiawei Han. A bayesian approach to discovering truth from conflicting sources for data integration. *Proc. VLDB Endow.*, 5(6), February 2012. 7.4, 7.4