Analytics Meta Learning

Zi Yang

March 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dr. Eric Nyberg, Carnegie Mellon University
Dr. Jaime Carbonell, Carnegie Mellon University
Dr. Ravi Starzl, Carnegie Mellon University
Dr. John M. Prager, IBM T. J. Watson Research Center
Dr. ChengXiang Zhai, University of Illinois at Urbana-Champaign

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

Copyright © 2016 Zi Yang
**Keywords:** Analytics Meta Learning, Procedure Space Exploration, Intelligent Information System, Question Answer, Decision Support
Abstract

Data analytics is the process to analyze massive data to discover useful knowledge and make conclusion about the information, to improve predictions and support decision making. Solving analytics tasks requires more than just merely applying analysis algorithms, instead it combines high-level decision making and low-level process execution, which makes it more difficult than performing individual analyses or analysis steps. As more and more analysis components have become available nowadays, it has been more challenging and time-consuming than ever to quickly design an extensible architecture, and effectively and efficiently compose an information system from these components in order to achieve a desired or optimal level of performance on a given analytics task. In this these, we study both theoretically and empirically the problem and the solution to assist, if not replace, humans the design, planning, and evaluation in the development of intelligent information systems for analytics tasks. We refer this problem as analytics meta learning.

In the thesis proposal, we formally define the problem of analytics meta learning and propose a solution framework that consists of three steps: analytics procedure definition, analysis component construction, and analytics space exploration. From the theoretical perspective, we focus on design and rigorous analysis of algorithms to extract procedural knowledge for analytics procedure definition and learn to optimize procedures from task benchmarks. We implement a software architecture framework that enables analytics meta learning, and we leverage the framework to solve real-world analytics tasks on three domains of problems, including general biomedical information seeking task, pharmaceutical decision support task, and product recommendation task, and empirically study the performance of the proposed algorithms.

To date, we have made initial achievements in one or some aspects of analytics meta learning. We have developed representation languages for analytics procedures (ECD and DPT), and proposed a greedy algorithm for optimizing a configurable information pipeline (CSE algorithm). The ECD framework has been successfully used in developing intelligent information systems across various domains since its first release in mid 2012, which includes TREC Genomics passage retrieval task, BioASQ QA task, target validation for drug discovery, etc. We have seen positive outcomes from a set of preliminary experiments, e.g., our BioASQ exact answer generation pipeline was ranked #1 in 5 out of 6 focused categories in the official evaluations. The framework has also been employed in the related courses domestically and abroad for educational purposes since 2012. The goal of the thesis is to formally define and systematically study the analytics meta learning problem through theoretical analyses and empirical case studies on a wide range of applications.
Acknowledgments
## Contents

1 **Introduction** .................. 1
   1.1 Thesis Statement .................. 3
   1.2 Thesis Proposal Overview ............ 4
   1.3 Initial Achievements ............... 5
   1.4 Thesis Proposal Structure ....... 8

2 **Analytics Meta Learning** ........ 13
   2.1 Related Work .................. 14
      2.1.1 Execution Process Representation .......... 14
      2.1.2 Decision Process Representation ............. 16
      2.1.3 Information Processing Architecture Frameworks ........ 17
      2.1.4 Limitations of Specification Languages ......... 19
   2.2 Terminology .................. 20
   2.3 Problem Definition & Solution Framework ....... 24
   2.4 Analytics Procedure Definition .......... 29
   2.5 Proposed Work .................. 29

3 **Procedural Knowledge Discovery** ........ 31
   3.1 Related Work .................. 31
      3.1.1 Instruction Extraction from Semi-Structured Text .......... 32
      3.1.2 Script Text Extraction .................. 32
      3.1.3 Causal Relation & Argumentation Extraction ............. 32
   3.2 Procedural Knowledge Base ........ 33
      3.2.1 Semi-Structured Procedural Knowledge Base ......... 33
   3.3 Application: Search Task Suggestion .......... 35
      3.3.1 Modeling & Solution ............. 36
      3.3.2 Data Preparation ............. 38
      3.3.3 Experiment Settings ............. 40
      3.3.4 Experimental Results ............. 41
   3.4 Procedural Knowledge Extraction from Search Activities ........ 43
      3.4.1 Modeling & Solution ............. 45
      3.4.2 Data Preparation ............. 47
      3.4.3 Experiment Settings ............. 48
3.4.4 Experimental Results .................................................. 48
3.5 Procedural Knowledge Extraction from Unstructured Text .......... 50
3.6 Procedural Knowledge Extraction from Community QA Site ........ 51
3.7 Proposed Work ............................................................... 51

4 Exploration Algorithms .................................................... 53
4.1 Related Work ................................................................. 53
  4.1.1 Automatic Service Composition ..................................... 53
  4.1.2 Limitations of Automatic Service Composition Solutions ........ 53
  4.1.3 Automatic Optimization of Configurable Systems ............... 54
  4.1.4 Optimization Theories ................................................ 54
  4.1.5 Automatic Evaluation of Software ................................... 54
4.2 Configuration Space Exploration ......................................... 55
  4.2.1 Solution Overview .................................................... 56
  4.2.2 Modeling cost and benefit distributions ......................... 58
  4.2.3 Optimization via stochastic scheduling .......................... 59
  4.2.4 An example: 3-stage linear characteristics ...................... 60
4.3 Preliminary Case Study: TREC Genomics Passage Retrieval ......... 61
  4.3.1 Task & data set description ........................................ 61
  4.3.2 Analysis component construction: resource and algorithm integration 62
  4.3.3 Experimental results ............................................... 63
  4.3.4 Component analysis ................................................ 64
  4.3.5 Exploration strategy performance analysis ...................... 67
  4.3.6 Generalizability analysis ......................................... 70
4.4 General Exploration Strategy ............................................ 71
  4.4.1 Modeling Decision Processes ...................................... 72
  4.4.2 Overall Optimization .............................................. 72
4.5 Proposed Work ............................................................... 73

5 Analytics Meta Learning Implementation ............................... 75
5.1 Principles of Design and Engineering ................................ 75
  5.1.1 Declarative Descriptors ............................................ 75
  5.1.2 Configurable Evaluation ........................................... 76
  5.1.3 Automatic Data Persistence ....................................... 76
  5.1.4 Configurable Exploration Strategy ............................... 77
  5.1.5 Global resource caching ........................................... 77
  5.1.6 Distributed Architecture .......................................... 77
  5.1.7 Graphical User Interface ......................................... 77
5.2 Open Source UIMA Extension ........................................... 77
  5.2.1 Proposed Work ...................................................... 78
# 6 Case Study: Biomedical Question Answering Task

## 6.1 Related Work

## 6.2 Task Description

### 6.2.1 Evaluation metrics

### 6.2.2 Dataset

## 6.3 Analytics Procedure Definition

## 6.4 Analysis Component Construction: Learning to Answer Biomedical Questions

### 6.4.1 Document Retrieval

### 6.4.2 Snippet Retrieval

### 6.4.3 Concept Retrieval

### 6.4.4 Triple Retrieval

### 6.4.5 Question and Answer Type Prediction

### 6.4.6 Candidate Answer Generation

### 6.4.7 Candidate Answer Scoring

### 6.4.8 Answer Pruning

### 6.4.9 Yes/No Evidencing

### 6.4.10 Ideal Answer Generation

## 6.5 Preliminary Results & Analysis

## 6.6 Analytics Space Exploration

## 6.7 Proposed Work

# 7 Case Study: Pharmaceutical Decision Task

## 7.1 Related Work

## 7.2 Task Description

### 7.2.1 Evaluation metrics

### 7.2.2 Dataset

## 7.3 Analytics Procedure Definition

### 7.3.1 Manual Analytics Procedure for Target Validation

### 7.3.2 Automatic Analytics Procedure Construction

## 7.4 Analysis Component Construction

### 7.4.1 Supporting Evidence based Yes/No QA Analysis Components

### 7.4.2 Adapting General Biomedical QA

## 7.5 Analytics Space Exploration

## 7.6 Proposed Work

# 8 Case Study: Product Recommendation Task

## 8.1 Task Description

### 8.1.1 Evaluation metrics

### 8.1.2 Dataset

## 8.2 Analytics Procedure Definition

### 8.2.1 Manual Decision Process for Buying a Phone

### 8.2.2 Automatic Analytics Procedure Construction

## 8.3 Analysis Component Construction
List of Figures

1.1 Use case diagram of data analytics ................................................. 2
1.2 Analytics meta learning problem lies in the intersection .................. 3
1.3 Architecture overview of the QUADS framework ............................ 7
1.4 Architecture overview of the CSE framework ................................. 8
1.5 Thesis structure ........................................................................... 11

3.1 Subproblems in search task suggestion using procedural knowledge and automatic procedural knowledge base construction from search activities .......... 45
3.2 An example of the parallel corpus construction process .................... 46

4.1 Modeling cost and benefit for analytics space exploration ................ 58
4.2 Variation of real-valued configurations for all components in the TREC Genomics experiment .............................................................. 66
4.3 Estimation of likelihood and posterior of cost at the end of the TREC Genomics experiment ................................................................. 68
4.4 Estimation of likelihood and posterior of cost when the TREC Genomics experiment completed 1% of total execution tasks .................. 69
4.5 Estimation of likelihood and posterior of component-level benefit when 20% of total execution tasks were completed in the TREC Genomics experiment .......................................................... 70
4.6 Generalizability of systems with various resampling methods in the TREC Genomics experiment ............................................................. 71

6.1 BioASQ QA Phase B pipeline diagram ............................................. 84
6.2 BioASQ QA Phase B pipeline diagram ............................................. 85

7.1 Decision subtasks in the three-level target validation task ................ 105

9.1 Gantt chart for the proposed work .................................................. 115
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison between supervised learning and analytics meta learning</td>
<td>28</td>
</tr>
<tr>
<td>3.1</td>
<td>Semi-structured procedural knowledge base feature definition</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Averaged number and percentage of related queries suggested by Google or Bing</td>
<td>40</td>
</tr>
<tr>
<td>or issued by the users subsequently in the same session that are mentioned by a task description in wikiHow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Search task suggestion results</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>The non-Word features that contribute the most to search task suggestion problem</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Search task suggestion results</td>
<td>43</td>
</tr>
<tr>
<td>3.6</td>
<td>Top 4 suggested search tasks from the proposed STS system and commercial search engines for example search tasks</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>Task summary generation results</td>
<td>48</td>
</tr>
<tr>
<td>3.8</td>
<td>Task explanation generation results</td>
<td>49</td>
</tr>
<tr>
<td>3.9</td>
<td>The non-Word features that contribute the most to the automatic procedural knowledge base construction problem</td>
<td>50</td>
</tr>
<tr>
<td>3.10</td>
<td>Summary and explanation generation result for automatic procedural knowledge base construction</td>
<td>50</td>
</tr>
<tr>
<td>3.11</td>
<td>Suggested subtasks from the proposed APKBC system for example tasks</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Posterior distributions for each of $\gamma_c$, $\gamma_t$ and $\beta_c$, $\beta_t$ in modeling analytics space exploration</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>Summary of analysis components in TREC Genomics QA system</td>
<td>62</td>
</tr>
<tr>
<td>4.3</td>
<td>Experimental settings and comparison with TREC Genomics 2006 participants</td>
<td>64</td>
</tr>
<tr>
<td>4.4</td>
<td>Variation of nominal configurations for all components in the TREC Genomics experiment</td>
<td>65</td>
</tr>
<tr>
<td>4.5</td>
<td>Result on TREC Genomics 2007 data set</td>
<td>71</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison of biomedical question answering shared tasks: TREC Genomics and BioASQ</td>
<td>86</td>
</tr>
<tr>
<td>6.2</td>
<td>BioASQ Phase B Snippet Retrieval Features</td>
<td>89</td>
</tr>
<tr>
<td>6.3</td>
<td>BioASQ Phase B Answer Type Prediction Features</td>
<td>93</td>
</tr>
<tr>
<td>6.4</td>
<td>BioASQ Phase B Answer Scorers</td>
<td>95</td>
</tr>
<tr>
<td>6.5</td>
<td>Partial official BioASQ 3B evaluation result and ranks among systems</td>
<td>97</td>
</tr>
<tr>
<td>6.6</td>
<td>Error categories and occurrences for factoid questions in BioASQ 3B</td>
<td>97</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Data analytics is the process to analyze massive data to discover useful knowledge and make conclusion about the information, to improve predictions and support decision making [78]. A specific data analytics use case can be defined by, at simplest, a task description representing the user’s information seeking or decision making need. The following questions are examples of real-life information needs that have been reported from biomedical experts during their research or diagnosis: [95,170] (Examples 1 to 5) and from real online shoppers (Example 6):

(Ex. 1) How many TAp73 isoforms have been identified in humans?
(Ex. 2) What is the function of the mamalian gene Irg1?
(Ex. 3) Find the most frequent carbapenemases found in Enterobacteriaceae.
(Ex. 4) Can Alzheimer’s disease related miRNAs be detected in patients’ blood?
(Ex. 5) Is AKT1 directly involved in the breast cancer and can be a suitable target?
(Ex. 6) Choose an Android phone that best suits your needs.

A data analytics procedural to answer these questions and make their own conclusion include discovering and evaluating all the relevant, but possibly inconsistent or even contradictory, conclusions from over 20 million biomedical research publications^[1]^ as well as various manually curated databases. Solving analytics tasks requires more than just merely applying analysis algorithms, instead it combines high-level decision making and low-level process execution, which makes it more difficult than performing individual analyses or analysis steps. As a result, in order to support users’ decision making, an intelligent information system that solves an analytics task usually requires a systematic design and implementation based on a model of human cognition, which usually consists of multiple internal processing steps and integrates various internal and external analysis steps, including algorithms, knowledge bases or other resources from heterogeneous sources, as well as other analytics engines for sub analytics tasks as subroutine. A use case diagram is depicted in Figure [1.1]. In addition to the primary use case that the user submits an analytics task to the system and await response, we further consider that the system can leverage human decision knowledge by working actively with the user or a domain expert to tackle the analytics task.

As more and more analysis components have become available nowadays, it has been more challenging and time-consuming than ever to quickly design an extensible architecture, and effectively and efficiently compose an information system from these components in order to achieve a desired or optimal level of performance on a given analytics task. First, since the modules might be developed by people of wide range of skills and abilities, experienced core team members, novices, or unmet open source contributors, their quality might vary. Furthermore, the interoperability between data analysis algorithms and resources is also hardly guaranteed by their original developers. In the meantime, the requirement and expected performance of the information systems keep changing as new information demands arise from their users.

Proposing practices and tools to guide and automate composition, optimization, and evaluation has been a focus of information system design and development. To support integration of data analysis components, information processing architecture frameworks, such as UIMA [41, 42], define an information system in terms of a set of basic elements (such as typed object, module, configuration, resource, workflow, etc.) described via standardized representation. Explicit specification of preconditions and effects has also enabled automatic composition of applications from components [129]. However, when we apply these approaches to constructing an information system for an arbitrary analytics task, most approaches would be possible to produce a functioning system, but not sufficiently intelligent to perform the analytics task, due to the limitations of expressiveness in the specification languages and the ignorance of analytics task objectives during optimization. Similarly, most test automation frameworks [102, 146] can only serve as a sanity checker for intelligent information systems, since their performance on a given analytics task is usually judged subjectively on a graded or continuous scale, rather than binary “pass or fail”.

We believe that there is great demand for a comprehensive solution to assist, if not replace, humans the design, planning, and evaluation in the development of intelligent information systems for analytics tasks, to guarantee continuous improvement towards to the task objective. In this thesis, we envision a software framework, independent of specific analytics tasks, that extends a
Figure 1.2: Analytics meta learning problem lies in the intersection of data analytics, software architecture and supervised learning.

The biggest challenge in the development of the intelligent architecture layer is to design an algorithm that learns arbitrary analytics procedures that combine lower-level analysis components and analytics, simultaneously as these analysis components and analytics are optimizing and updating their own models individually and locally. We refer this problem as analytics meta learning, in analogy to traditional learning problem in data analytics, where a predefined and configured training process is statically performed, when new training data arrive, to update a model. The proposed work lies in the intersection of data analytics, software architecture and supervised learning, as shown in Figure 1.2.

1.1 Thesis Statement

The thesis introduces the analytics meta learning methodology to standardize, accelerate, and automate construction and optimization of analytics procedures for arbitrary data analytics tasks. Through both theoretical studies of each respective problem – analytics procedure definition, analysis component construction, and analytics space exploration – and empirical studies using real-
world analytics tasks, we claim the proposed methodology and the algorithms can achieve the desired level of performance in order to assist, if not replace, human in the design, planning, and evaluation during the development of intelligent information systems for analytics tasks, to guarantee continuous improvement towards to the task objective.

1.2 Thesis Proposal Overview

We study the problem from both theoretical and empirical perspectives. We start the theoretical study with giving formal definitions to the subjects we study in the thesis, which include analytics task, analytics procedure, information system (analytics engine), etc. We then formally define the analytics meta learning problem and present the solution framework, which consists of three major steps: analytics procedure definition, analysis component construction, and analytics space exploration.

To facilitate the analytics procedure definition process during the development of intelligent information systems and overcome the limitations in existing execution and/or decision process specification languages, we introduce a unified procedure representation language, which allows cross reference between processes, since decision processes and execution processes may inter-sperse with each other in an analytics procedure for a complex analytics task. The unified procedure presentation also allows specification of alternative options, which creates the foundation for automatic construction and optimization of analytics procedures. Designing and describing analytics procedures, especially those that involve decision processes, is not an easy task, especially for the users who know little about the problem domain. As a subproblem of analytics meta learning, we introduce that problem of procedural knowledge discovery, which aims to automatically discover procedural knowledge – the knowledge exercised in the performance of some task, i.e. what actions should be performed (execution processes) and what factors should be considered to achieve some goal (decision processes) [6, 48] – and construct analytics procedures for arbitrary analytics tasks. In this thesis, we explore a wider range of human-generated data sources, such as on-line community question answering sites, bibliographic databases, etc., and study the canonicalization of representation to facilitate analysis component construction and execution. As oppose to domain/task-specific analysis component construction step, we study the general analytics space exploration problem, which operates directly on any arbitrary predefined and constructed analytics space. We review literature on automated service composition, planning and scheduling, and attempt to propose exploration strategies. There are also many theoretical questions that we intend to answer in the thesis, e.g. the rationality of the performance assumptions, the asymptotic property and optimality of the proposed approach, etc.

In the empirical or experimental part of the thesis, we take analytics meta learning into action. We focus on exemplifying the proposed methodology to tackle analytics tasks and reporting the performance of the proposed analytics meta learning method in terms of the prediction rate in each planning process and the learning rate throughout the entire learning process, as well as the performance of its resulting products, i.e. information systems, in terms of task-specific metrics and measurements. We first present the principles of design and implementation of analytics meta learning software framework, and accordingly develop an open source toolkit that supports the uni-
fied representation language and bundles the implementations to the various algorithms proposed in the theoretical studies. We then focus on three case studies: biomedical question answering task, pharmaceutical decision task, product recommendation task.

We first apply analytics meta learning to the BioASQ [154], a biomedical QA task. Since the task has predefined checkpoints at certain stages along the pipeline, we, in this case study, focus on analysis component optimization, rather than procedure construction and optimization. We develop analysis components for all functionalities, especially we investigate whether it is possible to design and train supervised models to answer questions, without the use of manually-constructed rules or predefined templates. We then proceed to leverage the analytics space exploration algorithms to boost the system performance using the BioASQ 2013–2015 benchmark set containing over 1K decision scenarios nominated and their gold standard solutions created by biomedical professionals. Next, we consider two complex analytics task scenarios that require human decision making – pharmaceutical decision task and product recommendation task, where we demonstrate construction and optimization of both analytics procedure and analysis components. We first perform the proposed automatic procedure construction algorithms to selected pharmaceutical decision tasks (e.g. target validation) and selected product recommendation tasks, and compare the outputs with human decision processes (e.g. target validation DPT). Then, we employ the general-purpose QA components and pipelines to perform sub analytics tasks. Finally, we explore the analytics spaces that contain massive analytics procedures, where we also investigate how analytics meta learning advances rapid domain adaptation, i.e. given a system pre-optimized for an existing analytics task (e.g. BioASQ task), how we can quickly adapt the system to a related but different analytics task (e.g. drug discovery).

1.3 Initial Achievements

To date, we have done some preliminary study in one or some aspects of analytics meta learning, including representation languages for analytics procedures, a greedy algorithm for optimizing a configurable intelligent information system, a supervised decision synthesis algorithm, a procedural knowledge extraction method for decision support. We have also developed analysis components and pipelines for a number of real-world analytics tasks, which serve as use cases for empirically studying analytics meta learning. In this section, we focus on presenting our initial achievements.

First, we proposed a YAML-based execution process description language, Extended Configuration Descriptor (ECD) [169], which extends the description languages in the UIMA framework. ECD language inherits many advantages of the description language in the UIMA framework, e.g. a multi-layered design that separates the component-level Analysis Engine (AE) descriptors from the pipeline-level Collection Processing Engine (CPE) descriptors, an explicit distinction between the capabilities and configuration parameters, etc. Furthermore, it provides a compact and editable representation of an entire solution space for an analytics task by all possible configurations of all analysis components from which an intelligent information system can be composed. We also

1.3 Initial Achievements

To date, we have done some preliminary study in one or some aspects of analytics meta learning, including representation languages for analytics procedures, a greedy algorithm for optimizing a configurable intelligent information system, a supervised decision synthesis algorithm, a procedural knowledge extraction method for decision support. We have also developed analysis components and pipelines for a number of real-world analytics tasks, which serve as use cases for empirically studying analytics meta learning. In this section, we focus on presenting our initial achievements.

First, we proposed a YAML-based execution process description language, Extended Configuration Descriptor (ECD) [169], which extends the description languages in the UIMA framework. ECD language inherits many advantages of the description language in the UIMA framework, e.g. a multi-layered design that separates the component-level Analysis Engine (AE) descriptors from the pipeline-level Collection Processing Engine (CPE) descriptors, an explicit distinction between the capabilities and configuration parameters, etc. Furthermore, it provides a compact and editable representation of an entire solution space for an analytics task by all possible configurations of all analysis components from which an intelligent information system can be composed. We also
developed a open source software toolkit to instantiate a family of information processing systems specified by an ECD and execute them in an effective manner. The ECD framework has been successfully used in developing intelligent information systems for various analytics tasks since its first release in mid 2012, e.g.

- (2011 – 2013) a biomedical document and passage retrieval system [169] that addresses the information needs in genomics research as defined in the TREC Genomics task [59], where we selected a number of biomedical knowledge bases, algorithms and toolkits for information retrieval, Natural Language Processing (NLP) and Machine Learning (ML) that were most cited by the task participants in their notebook papers and subsequent publications,

- (2013) a machine reading comprehension system [118] that is challenged by multiple-choice questions, as defined in the CLEF QA4MRE tasks [119], which covers various scenarios, e.g. reading comprehension tests in college entrance exams, Alzheimer’s disease treatments, etc.,

- (2013 – 2014) a general decision support framework, QUADS [170], which has been applied to two real-world applications: target validation, a fundamental decision-making task for the pharmaceutical industry, and product recommendation from review texts, an everyday decision-making situation faced by on-line consumers,

- (2014 – 2015) a biomedical QA system that complies with the requirement of the CLEF BioASQ QA task [154], which, compared to prior similar tasks, covers a wider range of biomedical subdomains and question types, and requires a more comprehensive decision report, including relevant documents, snippets, concepts, RDF triples, in addition to the standard exact answers and summaries. Our exact answer generation pipeline was ranked #1 in 5 out of 6 focused categories in the official evaluations.

We have also taught the ECD framework in a number of graduate-level courses related to design and engineering of intelligent information systems, and employed it as the development and collaboration platform in the course projects, which include

- (Spring, Fall 2012) CMU 11-634/11-632: MCDS Capstone Project / MCDS Capstone Planning Seminar,
- (Fall 2012, 2013) CMU 11-791/11-792: Software Engineering for Information Systems / Design and Engineering of Intelligent Information Systems,
- (Spring 2014) ITAM COM-35702: Sistemas Inteligentes,
- (Spring 2013, 2015) CMU 11-797/11-796: Question Answering / Question Answering Lab,
- (Spring, Fall 2015) CMU 11-697/11-696: MIIS Capstone Project / MIIS Capstone Planning Seminar.

In another effort to provide a more effective medium to describe arbitrary analytics task solution space, we proposed a YAML-based decision process description language, Decision Process Template (DPT) [170]. To overcome the inflexibility in adapting to new decision scenarios of traditional decision support systems [21][144], a novelty of DPT is that it allows users to describe the task goal and steps using natural language. In addition to specify an execution process (i.e. ECD) that directly tackles the task, DPT also allows to specify analytical thinking as sub-processes by giving references to other decision factors (also DPTs) that should be considered for the decision
scenario, which captures non-linear human decision logics, such as trees or graphs. Following the DPT guidelines, we created a target validation DPT, based on literature review as well as personal knowledge and experience from a group of professional biochemists and bioinformaticians working at a pharmaceutical company, and a cell phone recommendation DPT. The QUADS framework [170], as depicted in Figure 1.3, could automatically understand and process the decision scenario via DPT in real time, providing an overall weighted decision by combining evidence from answers to individual questions. We applied the QUADS framework to the two applications to help bioinformaticians validate potential gene targets for diseases and help on-line customers to choose products based on users’ reviews.

To enable automatic construction and optimization of execution processes, we developed the Configuration Space Exploration (CSE) framework [169] based on the ECD framework, as shown in Figure 1.4, with two notable extensions: performance evaluation and data persistence. Beyond the brute force approach that exhaustively evaluates all the pipeline configurations, in our initial study of the analytics meta learning problem, we proposed an exploration strategy based on hierarchical Bayesian modeling and stochastic scheduling to improve the overall system performance through iterative update of the pipeline configuration. The CSE framework with the built-in exploration strategy was first used to optimize the above-mentioned biomedical document and passage retrieval system over a trillion different configurations of components and parameter values. The results have suggested that the proposed solution was able to find an optimal configuration of components for TREC Genomics [59] with a passage MAP score better than prior published results for the task [169]. At the same time as we tested the CSE framework with a real-world analytics task – TREC Genomics task, a group of students taking the Question Answering course were developing
the machine reading comprehension system for the upcoming CLEF 2013 QA4MRE challenge [119]. Throughout the semester, each student was individually developing one or some of the analysis components and further optimizing these modules by fixing the rest of the components unchanged from the baseline system. One day prior to the submission deadline, we managed to convert the vanilla UIMA descriptors to ECDs, and then leveraged the CSE framework to identify a better combination, which achieved 59.6% performance gain over the system that integrated all the individually optimized components [118].

Until recently, either the user or an expert had had to manually and explicitly specify the general human decision processes via DPT, such as the target validation DPT and the cell phone recommendation DPT, before the user leveraged the QUADS framework for decision support. Most recently, we made a first attempt to study procedural knowledge discovery from heterogeneous sources, and eventually construct DPTs automatically without human interference. We proposed to take advantage of an existing procedural knowledge base, e.g. wikiHow [3] to assist task-oriented information seeking tasks, using query suggestion as an example [168]. We compare our proposed solution with baseline algorithms, commercial search engines, and the (manually-curated) wikiHow procedural knowledge; experimental results show an improvement of +0.28 to +0.41 in terms of Precision@8 and Mean Average Precision (MAP).

1.4 Thesis Proposal Structure

The thesis proposal is organized as follows:

Chapter 2 Analytics Meta Learning

This chapter first formally defines the terminology related to analytics meta learning, in-
cluding analytics tasks, analysis components, information systems, etc., then introduces the problem of analytics meta learning, and presents a general solution framework that consists of analytics procedure definition, analytics component construction, and configurable space exploration. In the last section of this chapter, we focus on analytics procedures representations. We first present the ECD and DPT representations for execution processes and decision processes, and then propose a unified presentation for analytics meta learning, which allows cross reference between processes and specification alternative options.

Chapter 3: Procedural Knowledge Discovery
Despite manual specification of analytics procedures as described in Chapter 2, this chapter focuses on a subfield of analytics meta learning – automatic procedure discovery algorithms. We explore a wide range of human-generated data sources and design algorithms to mine such knowledge and canonicalize the representation from each respective source. We first focus on three types of data sources that we have successfully leveraged to this problem including users’ search query log, semi-structured human curated procedural knowledge bases, and relevant Web documents from search engine result page (SEPR), etc., and then propose methods to explore additional sources, such as on-line community questions answering sites, bibliographic databases, etc.

Chapter 4: Exploration Algorithms
This chapter studies the optimization problem in analytics meta learning – analytics space exploration, which aims to rapidly identify the optimal analytics procedures. We first present an exploration strategy based on hierarchical Bayesian model and stochastic scheduling, and report the experimental results on our preliminary studies using the TREC Genomics QA task, which involves a configuration space (a subspace of an analytics space) consisting of trillions of possible execution processes. Based on the literature review on overflow scheduling and planning, we propose additional exploration strategies, and conduct a theoretical comparison across these strategies, such as the rationality of their assumptions, the asymptotic property and the optimality of the optimization algorithms, etc.

Chapter 5: Analytics Meta Learning Implementation
This chapter focused on the development of the analytics meta learning framework. We discuss the high-level design and engineering principles of the architecture layer for those who intend to implement for their in-house architecture frameworks other than UIMA, and describe our current implementation based on Apache UIMA and uimaFIT. We also present the development plans to realize all the desired functionalities.

Chapter 6: Case Study: Biomedical Question Answering Task
This chapter presents the use of analytics meta learning in construction of a BioASQ-like information system that supports a series of real-life information seeking needs from biomedical experts. We first define the task and present the analysis components that we have developed for the BioASQ 2015 challenge, then we detail the development plan for the missing functionalities, such as yes/no QA, summary generation, etc., to complement the current system, as well as the experiment plan for analysis component and execution process

\[https://uima.apache.org/uimafit.html\]
optimization using the proposed exploration strategies.

Chapter 7: Case Study: Pharmaceutical Decision Task

This chapter presents the use of analytics meta learning for complex pharmaceutical decision scenarios. We describe the analysis components that we have developed for the target validation support task, e.g. yes/no QA components, decision synthesis components, and we report the initial results using the QUADS framework that incorporates a manually created DPT. We propose to apply a full analytics meta learning approach to construct and optimize a general-purpose pharmaceutical decision support system. We propose to take advantage of biomedical documents to extract human decision processes for various decision scenarios, and then outline the necessary steps to conduct a comprehensive exploration of all possible procedures and configurations across components.

Chapter 8: Case Study: Product Recommendation Task

This chapter focuses on another use case that involves complex problem solving procedure – product recommendation task. We describe the analysis components that we have developed for the phone recommendation task, and we report the initial results using the QUADS framework that incorporates a manually created DPT. We propose to apply a full analytics meta learning approach to construct and optimize a general-purpose product recommendation system using review texts and product databases. We propose to take advantage of purchase guide documents to extract human decision processes for various decision scenarios, and then outline the steps to conduct a comprehensive exploration of all possible procedures and configurations across components.

Chapter 9: Conclusion and Future Work

This chapter makes the conclusions and presents the time line for the proposed work.

We summarize the structure in Figure 1.5.
We introduce a general solution framework that extends UIMA (detailed in §2.3).

**Unified representation**
Persistence and process of the unified procedure representation (detailed in §2.4).

**Procedural knowledge discovery**
Automatic discovery of analytics procedures (detailed in §3).

**Analytics space exploration**
Automatic construction and exploration of analytics spaces (detailed in §4).

**Pharmaceutical decision support**
A decision support system using Bio QA technology in drug discovery, detailed in §7.

**Pharmaceutical QA**
A QA system adapted to drug discovery that focuses on availability of the components, detailed in §7.4.

**TREC Genomics**
A document / passage retrieval system related to genomics (detailed in §4.3).

**CLEF QA4MRE**
A reading comprehension system evaluated using multiple-choice questions (detailed in §4.3).

**CLEF BioASQ**
A general-purpose biomedical QA system for information seeking and decision support (detailed in §6).

**Product analytics engine**
A retrieval and QA system focusing on product related information (detailed in §8.3).

**Biomedical analysis components**
(detailed in §6.4).

**General analysis components**
(detailed in §8.3).

**Analytics meta learning framework**
We introduce a general solution framework that extends UIMA (detailed in §2.3).

**UIMA**

**JAVA**

Figure 1.5: Thesis structure
Chapter 2

Analytics Meta Learning

Researchers have studied related problems from various angles. For example, operations researchers and decision scientists have focused on describe and formalize decision making problems and solutions [63, 133], while NLP and IR researchers have developed pieces of software to help understand human decision needs and collect relevant information. Software researchers have often viewed the problem from a systematic perspective, and proposed software development methodologies [13, 111], automatic service composition algorithms [129] and automatic evaluation of systems [102, 146]. The goal of analytics meta learning is to bridge the gaps between these related research areas and provide a comprehensive solution that assists developers in developing information systems given arbitrary analytics tasks.

In the next three chapters, we focus on presenting the theoretical foundation of the analytics meta learning problem, where we try to answer a series of questions, e.g. what analytics meta learning is, how it relates to traditional machine learning problems in data analytics, how we design algorithms to solve the analytics meta learning problem, and what the steps are when we want to solve an analytics task using analytics meta learning.

This chapter presents the general setup for the analytics meta learning. We first review prior work related to representation and standardization of information processing pipelines in Section 2.1 while we leave the literature review related to automatic discovery and optimization of analytics task workflows in Sections 3.1 and 4.1. Since we found that researchers could express the same concept in various ways, and on the other hand, the same expression could have different meanings in different contexts, e.g. goal vs. target vs. purpose, instruction vs. action sequence, step vs. action, etc., in Section 2.2, we give formal definition of terminology used in the thesis, such as analytics task, analysis component, information system, analytics procedure, etc., and introduce mathematical notations to each of them for ease of reference later in the thesis. Then, in Section 2.3, we formally define the problem of analytics meta learning, and presents a general solution framework that consists of three major steps: analytics procedure definition, analysis component construction, and analytics space exploration. In Section 2.4, we focus on formal analytics procedure representation. We first define representation schemata for execution processes and decision processes, and then propose a unified presentation for analytics meta learning, which allows cross reference between processes and specification alternative options.
2.1 Related Work

The problem we study in the thesis is motivated by the great demand for assisting and automating the design and development of intelligent information systems. In order to effectively and efficiently compose an information system for a given analytics task, developers tend to leverage existing analysis components developed by people of wide range of skills and abilities, experienced core team members, novices, or unmet open source contributors. However, the data analysis algorithms and resources can hardly guarantee the interoperability between them; still less can they be reasoned and manipulated. In this section, we see that it is the description languages that drive the evolutionary progress towards automatic development of intelligent information systems.

To support integration of data analysis components, *de facto or de jure, proprietary or open* standards have been introduced to achieve interoperability at different levels. Data exchangeability is a primitive requirement for component interoperability. For example, the JAVALIN system [112] introduces a shared data standard via XML schema, and each component in the system deserializes an XML data stream for processing and serializes the output data into an XML stream, but how each component processes data is unclear at the data standard level. To further enable component and configuration interoperability, an Application Programming Interface (API) is often defined to characterize the inputs, outputs, parameters and functionalities of each component, where specification languages, e.g. XML-based languages such as, WSDL-S [4], OWL-S [54, 76] and Analysis Engine XML Descriptor [152], are also leveraged to describe components rather than data.

Description languages have later been utilized to represent workflows that combine individual components. In the next subsections, we review the representations of workflows (execution processes) of an information processing pipeline and decision processes when solving analytics tasks, and then information processing architecture frameworks that provide a comprehensive solution to manage the components.

2.1.1 Execution Process Representation

In this thesis, an execution process representation may refer to either an abstract symbolic representation for problem modeling (also referred to as *process modeling languages*), a machine-readable representation that combines machine executable subprocesses, or a human-friendly unstructured or semi-structured step-by-step manual written in natural language. Compared with the first two types, the last representation has recently gained much attention due to the expansion of user generated contents on the Web. One objective of the thesis is to try to establish its equivalence with the former two.

**Abstract symbolic representation.** Researchers have leveraged graph representations and graph theories to model execution processes, or business processes in general, including published standards, e.g. Business Process Definition Metamodel (BPDM) [50], Business Process Model and Notation (BPMN) [51], as well as academic proposals, e.g. Event Driven Process Chain (EPC) [138], Integrated DEFinition Method 3 (IDEF3) [96], Petri Net [109, 123], Role Activity Diagram (RAD) [62], etc. Among them, BPMN [51] was first released in 2004 and since then has been maintained by the Object Management Group (OMG). Flow objects are the main describing elements within BPMN, and consist of three core elements: *events, activities, and gateways*, where
activities can represent tasks, subprocesses, etc, and gateways determine forking and merging of paths. Compared to BPMN and other industrial standards, Petri nets [109, 123], which mainly consist of place and transition nodes to represent possible states of the system and events or actions causing state changes, have an exact mathematical definition of their execution semantics, with a well-developed mathematical theory for process analysis, which therefore have been utilized for mathematical proofs of process behaviors (e.g. in [123]). List and Korherr [88] conduct a comparison between these conceptual modeling languages. Researchers have also introduced representations for processes with special properties, e.g. Workflow nets [156], a subclass of Petri nets [109, 123] specifically designed to represent workflow procedures, and Refined Process Structure Tree (RPST) [159], which is equivalent to the process models where the process components are canonical. In this thesis, we also focus on (well-)structured process models [159], whose definition is given in Section 2.7, while research has also been done to proof the equivalence and convertibility between the processes using the abovementioned representations such as BPDM, Petri Nets, RPST, etc [113, 125].

**Machine-readable representation.** Various machine-readable execution process representation languages have been introduced, mainly from the software engineering and system management communities, to integrate different types of components. An example of such a language that interacts with external entities through Web service operations defined using WSDL is Business Process Execution Language (BPEL) [5], an OASIS standard for specifying actions within business processes with Web services. Beyond the basic processes, e.g. receive and reply, it also introduces structured processes to define control flows, such as sequence, flow, while, etc. Due to the distributed nature of Web services, BPEL is often used in a distributed or grid computing environment. Another example is XML Process Definition Language (XPDL) [24]. It focuses on the design, rather than the execution, of workflows, i.e., it cannot execute the processes or even has no clue how to execute them. Instead, it represents the interchanges of business process definitions between different workflow products, such as design tools, simulators, execution engines and Business Process Management (BPM) related tools. Aggregate Analysis Engine Descriptors, as part of the UIMA standard, can also be viewed as an execution process representation, which defines subprocesses via delegateAnalysisEngineSpecifiers, and manage the execution flow via flowController. Converting abstract conceptual representation to machine-readable representation has been extensively studied, such as BPMN to BPEL [114]. Although people have been argued about the applicability and limitation of various business process languages [157], we believe these languages still provide valuable insight and guideline how to describe and standardize execution processes.

**Human-friendly representation.** Cognitive psychology defines “procedural knowledge” (or imperative knowledge) as the knowledge exercised in the performance of some task [6, 47]. The Semantic Web community has attempted to formally define ontologies to represent procedural knowledge [17], which usually include an instruction (or action sequence) and a purpose (or goal). Also defined in such ontologies are the relations between procedural knowledge elements, at different levels of granularity, such as has-step, has-goal [116] or is-achieved-by [46]. Different from

In fact, a configuration parameter in the Flow Controller may refer to a description of the desired flow in some flow language such as BPEL.
machine-oriented execution process representations, procedural knowledge representations usually reflect business rules or best practices in everyday life and intend to share across experts and novices. For example, cooking recipe is a common type of execution process representation that demonstrates how to prepare or make a culinary dish. Installation instruction in the user’s manuals is another type of execution process representation. Similar to the machine-oriented execution process representations, the steps (equivalent to the primitive or lower level processes) and the order (equivalent to the flow) are the two important ingredients in the representation.

The analytics meta learning problem and our solution in the thesis are general and should work with arbitrary execution process representation and management tools. We give our definition to some of these concepts and discuss the connection to the prior models in Section 2.2, and then we further define concepts that are related to analytics meta learning that do not exist in the prior literature, e.g. procedure space. Also, the thesis attempts to bridge the gap between the abstract symbolic and machine-readable representations and human-friendly natural language representations, in order to construct an executable pipeline from a natural language objective.

### 2.1.2 Decision Process Representation

Although decision logic is an important part of procedural knowledge for many decision scenarios, esp. those that involve selection among several alternative possibilities, e.g. buying guide, career opportunity evaluation, etc., decision making and decision process representation are more often studied beyond the context of software engineering and system management. Therefore, there is another body of research to study the description and modeling of repeatable decisions within organizations to ensure that decision models are interchangeable across organizations. Again, we review abstract symbolic representations and machine-readable decision models that allow automatic validation and/or execution.

**Abstract symbolic representation.** In operations research or decision science, decisions are usually represented by a hierarchy or network of sub-criteria or sub-problems that are more easily comprehended or evaluated in isolation. Example representations include traditional decision table \[150\], decision tree, as well as Analytic Hierarchy Process (AHP), or Analytic Network Process (ANP) representation \[133\], or more compact Influence Diagram (ID, or decision diagram) \[63\], etc. AHP or ANP decomposes a decision problem into a hierarchy or network of more easily comprehended sub-problems, each of which can be analyzed independently. The elements of the hierarchy can relate to any aspect of the decision problem. Since estimating criteria weights and option priorities in AHP mostly relies on the judgments of human decision makers, pairwise comparisons of criteria and options are used in AHP instead of direct allocation of weights. AHP focuses on one single uncertainty with multiple alternatives, and decision makers are not allowed to explicitly assign priority judgments to non-primitive criteria. In contrast, an ID, an extension of a Bayesian network of chance nodes augmented with decision nodes, utility functions specifying the preferences of the decision maker, and a precedence ordering, can more flexibly model complex decision situations with multiple alternatives from multiple uncertainties, and allows the decision makers to specify their preference at any node in the diagram. However, finding the optimal solution to a general ID is much more computationally expensive than AHP, based on standard Bayesian inference technique \[67\]. Efficient exact and approximate algorithms for solving traditional decision
diagrams, Discrete Influence Diagrams (DID), Limited Memory Influence Diagrams (LMID)\cite{75, 90, 149}, etc., have therefore been extensively studied.

**Machine-readable decision models.** Object Management Group (OMG) has recently initiated an effort to establish a standardized specification of decision making with the introduction of the Decision Model and Notation (DMN)\cite{52}, which is considered complementary to the current BPMN standard for more efficient description of processes. DMN introduces four types of element at the decision requirements level: *input data, decision, business knowledge, knowledge source*, and the Friendly Enough Expression Language (FEEL) for specifying detailed decision logic.

In analytics meta learning problem, we give “first-class citizenship” to both execution processes and decision processes required for accomplishing an analytics task, which makes it possible to move towards cognitive computing – combining human expertise with machine components and achieving overall analytics meta learning. For the purpose of building information systems for analytics tasks, the decision requirements are considered part of the analytics procedure (together with execution processes), whereas the decision logics are specific algorithms and thus analysis components. Therefore, at the processes level, our analytics meta learning model combines the execution processes (as in BPMN) and decision requirements (as in DMN).

### 2.1.3 Information Processing Architecture Frameworks

As opposed to the execution process management tools that facilitate the developers in the *ex post* integration stage, information processing architecture frameworks, such as UIMA\cite{41, 42}, GATE\cite{26, 27}, Heart of Gold\cite{136}, Ellogon\cite{122}, TIPSTER\cite{49}, etc., on the other hand, manage the entire software development process in a principled way, from specification of requirement, design of system architecture, development of components, re-assessment and selection of features, functionalities or alternative solutions in each increment and iteration, testing, to final deployment of the complete system. In the development of Watson, the UIMA architecture enabled an average of roughly 25 core researchers and engineers to routinely prioritize, develop, and test new algorithms once every two weeks throughout the four-year development\cite{44}.

Information processing architecture frameworks are a suite of toolkits that standardize information systems in terms of a set of basic elements (such as typed object, module, configuration, resource, workflow, etc.). The Unstructured Information Management Architecture (UIMA)\cite{41, 42}, which is considered the most evolved and comprehensive architecture available\cite{10}, was originally developed by IBM to facilitate the analysis of unstructured information, including but not limited to text, speech, images or videos. It became an Apache Incubator project in 2006 before it graduated in 2010, and became an OASIS standard in 2009, and IBM’s analytics engine Watson\cite{43, 44} is a prominent example that uses the UIMA architecture. The UIMA framework defines the following elements to characterize information processing workflows:

- The artifact being analyzed (e.g., a document, audio file, video stream etc.) and the analysis results are encapsulated in a *Common Analysis Structure* (CAS), which provides a mechanism for shared access across analysis components.

\(^2\)An interesting comparison between these frameworks is described in a survey paper\cite{10}.  

17
• The schema or class model for the CAS is represented by a type system, which defines the types of objects and their properties (or features) that may be instantiated in a CAS.

• Analysis components interact (read, write, modify, etc.) with CAS by implementing the process method. Each analysis component can be associated with one or more Component Descriptors that contains metadata describing the component, its identity, structure and behavior. Configuring UIMA components is generally achieved by creating XML descriptor files, or using Java annotations directly in the component code via uimaFIT. An Analysis Engine (AE) can be instantiated from a component descriptor by the UIMA framework.

• A simple or primitive AE contains a single annotator. Aggregate Analysis Engines may be defined to contain other AEs organized in a workflow, where a flow specification defines the order in which the internal component AEs should be run.

The General Architecture for Text Engineering (GATE) \[26, 27\], which, despite its name, is also capable of processing audio-visual content, is another information processing architecture framework. Since the GATE framework only focuses on building Java-based information pipeline, it reuses many Java native features. All GATE resources are Java Beans. We briefly describe the GATE component model – Collection of REusable Objects for Language Engineering (CREOLE), in analogy to the UIMA framework.

• The objects being processed and analyzed, regardless of their granularity, lexicons, documents, or corpora, are all called Language Resources (LR). AnnotationSet is a data structure to store annotations specific to the Document type.

• Algorithmic components are named Processing Resources (PR), which can be configured similarly using XML descriptors or Java annotations.

• PRs can be grouped into Applications using controllers, where the controllers are also PRs, and thus they can be nested in another.

Measurability and reproducibility are also the focuses of information system architecture frameworks. Information retrieval communities have also held a series of competitions and conferences to create standard benchmarks and metrics to evaluate retrieval systems, and organizers have attempted to tabulate and analyze the most important features of high-performing systems, based on the information provided by the system reports, e.g. \[131\]. Due to lack of a standard task framework, organizers rarely reproduce experiments to try different combinations of modules across participants.

Recently, several shared tasks employ standardized data flows for successive subtasks (e.g. IR4QA and CCLQA tasks in the NTCIR Advanced Cross-Lingual Information Access task \[106, 107\]), and the results of the upstream tasks were collected from each system and shared for processing by all participants in the downstream tasks. This approach was able to discover configurations that combined different task systems to achieve a better result than the originally reported systems, which lends further motivation to configuration space exploration. However, the evaluation achieved interoperability through task-specific protocols, and did not provide a general framework for describing and evaluating configuration spaces that included variation on components and parameter settings.

To facilitate system evaluation based on commonly-shared benchmarks, researchers have re-
cently paid more attention to open and public evaluation infrastructures for information retrieval evaluation and information extraction [165]. More recently, the Workshop on Data infrastructureEs for Supporting Information Retrieval Evaluation [3] was held to focus on organizing benchmark collections and tools into coherent and integrated infrastructures to ensure reproducible and comparable experiments; various retrieval evaluation frameworks were presented [38]. However, systems are usually specific to a single retrieval task, and lack a clear separation of framework, component configuration, and component logic, which makes it hard to fully leverage existing tools through automatic configuration space exploration.

In this thesis, we study the theoretical problems in analytics meta learning regardless of any particular information processing architecture frameworks. In Chapter 5 we describe our extension to the UIMA framework for the architecture layer that achieves analytics meta learning. Our experience can be used to extend other existing open or proprietary information processing architecture frameworks.

2.1.4 Limitations of Specification Languages

These above-mentioned specification languages have too limited expressive power to meet the requirement of analysis components and intelligent information systems, e.g.

**Lack of expressivity of module configuration.** Many analysis components expose configuration parameters to allow easy adaptation [1]. For example, a pretrained model should be specified for a statistical NLP parser, and scaling coefficients $k_1$ and $b$ should be specified for Okapi BM25 [68] based retrieval component. Most current process description languages make no distinction between the configuration parameters and the actual input data to be processed and analyzed.

**Lack of expressivity of human decision logics.** Existing description languages focus on execution processes within analytics, rather than higher-order nonlinear decision processes that combine analytics and less complex analytical thinking as sub-processes. For example, solving analytics task (5) in the earlier example (“is AKT1 directly involved in the breast cancer and can be a suitable target?”) may include an execution process of searching for direct evidence from the literature and/or a decision process that considers factors (5.1) to (5.3). The description should be able to satisfy these new requirements, e.g. indicating what role each decision factor plays in solving the bigger problem, how the output from each decision factor should be synthesized, etc.

**Implicit unstated expectations.** There may exist implicit, unconscious or unstated expectation – how the components should be put together to achieve its maximum potential [1]. Developers may not become aware that such expectation exists, or they are not able to detail the actual semantics using existing specification ontology. This happens more often in building intelligent information system. For example, an Answer Generation (AG) component extracts named entities of the answer type from a list of passages generated by an upstream Passage Retrieval (PR) component. If the development team has effectively controlled the quality of the PR component, the AG developer may never need to consider the situation when a PR component also generates irrelevant passages, nor specify that the input passages should
reach a certain level of relevance as a precondition.

2.2 Terminology

In this section, we give formal definitions and notations to the prerequisite concepts needed for the definition of analytics meta learning, which include those that are commonly used in information system development and data analytics, e.g. analytics task, analytics procedure, analysis component, information system, etc. The terminology is introduced by following the same procedure that developers build an intelligent information system for an analytics task – we start with the description of an analytics task, and then break down into fine-grained analytics processes via a design process, create analysis components, and finally construct the information system via an instantiation process.

Definition 1 (Analytics, analytics task). **Data analytics** (or analytics) is the process (referred to as analytics procedure, denoted by \( p \)) to analyze massive data to discover useful knowledge about the information and make conclusion about a topic, to improve predictions and support decision making. Each analytics procedure \( p \) has a domain \( D(p) \) and a target \( T(p) \). An **analytics task** is a concrete information seeking or decision making need that can be achieved by performing analytics procedure \( p \), which is usually described by a **task description** that contains the input \( x \in D(p) \) representing the task requirement and the output \( y \in T(p) \) representing the expected outcome. The target set \( T(p) \) is often referred to as the **alternative set** in decision making if it is predefined.

Extracting input type and expected output type from natural language task descriptions is similar to the focus and answer type extraction in question analysis [79]. We take Task Example 1 in Chapter 1, the input, i.e. the information need, is “TAp73 isoforms have been identified in humans”, and the implicitly stated output type is a number that corresponds to the count of TAp73 isoforms. Benchmark dataset is an important source of task description with known ground truth outputs.

There are numerous kinds of analytics procedures of various complexity. A trivial form of an analytics procedure is a primitive step that can be executed with a single analysis component, which is referred to as **analysis step**. Usually the task descriptions are identical to the analysis step descriptions. For example, if the analytics task is defined as “to find all the words in a sentence”, then a whitespace-based tokenization, as an analysis step, can be considered as the analytics procedure to solve the task. Another form is a **linear-chain** execution process, which only comprise of consecutive execution steps in series. For example, Task Examples 1 to 3 in Chapter 1 may be addressed a typical factoid question answering pipeline, which consists of four main steps: question analyzer, document retriever, passage extractor, answer generator [151]. A typical ontology-based information extraction pipeline will integrate several preprocessors and aggregators [165]. We formally define execution processes as follows.

Definition 2 (Execution process). An analytics procedure \( p \) for analytics tasks \( D(p) \rightarrow T(p) \) is an **execution process** if it contains an ordered sequence of sub procedures: \( p_1, \ldots, p_n \) focusing on individual **subtasks** \( D(p_i) \rightarrow T(p_i) \) for each \( i = 1, \ldots, n \) such that
\[ D(p) \subseteq D(p_1), \]
\[ T(p_i) \subseteq D(p_{i+1}) \text{ for } i = 1, \ldots, n - 1, \text{ and} \]
\[ T(p_n) \subseteq T(p). \]

Many times, solving an analytics task also involves collecting and synthesizing indirect evidence for various factors that impact the overall decision, which requires human decision knowledge, which may differ across the stakeholders and evolve over time. For such complex multi-criteria analytics tasks, analytics may also employ a hierarchical decision process to capture tree-structured decision logics, such as with decision tree, decision table network \[150], or AHP \[133]. It follows the divide-and-conquer approach via problem reduction and solution synthesis phases. In the problem reduction phase, analytics procedures are often formalized by a recursive procedure that decomposes the task into a hierarchy or network of lower-level sub-criteria or sub-problems that are more easily comprehended and/or evaluated independently by sub analytics procedures \[105]\. In the solution synthesis phase, the original higher-level analytics task objective is achieved by synthesizing the lower-level conclusions obtained from sub analytics procedures.

For example, to make decision for Example 5, a number of sub analytics tasks, as listed in the following Examples 5.1 to 5.3, are considered necessary from the best practice of biomedical researchers \[65, 77, 170]\:

(Ex. 5.1) Is there any experiment showing that modulating the activity of the AKT1 with a chemical compound or genetic modification causes the breast cancer?

(Ex. 5.2) Is any mutation associated with the breast cancer?

(Ex. 5.3) Does any evidence suggest that targeting the AKT1 will have side effects?

Similarly, online how-to manuals, such as wikiHow\[3\] eHow\[4\] and many other task-specific guides, provide comprehensive and up-to-date instructions how to accomplish everyday tasks. For example, the following steps are suggested for Example 6 by a wikiHow article\[5\]:

(Ex. 6.1) Select a cellular carrier that is currently carrying Android phones.

(Ex. 6.2) Decide on the size and quality of the screen.

(Ex. 6.3) Read customer and professional reviews to get a better idea of the potential problems with any given Android phone you are considering.

We give formal definition to problem reduction and solution synthesis for the most general cases below.

**Definition 3** (Problem reduction, solution synthesis). Given all the analytics procedures \(p_1, \ldots, p_n\) and an analytics procedure \(p\), the problem reduction \(\rho_p\) is a function defined on \(\mathbb{R}^{D(p)} \rightarrow \mathbb{R}^{D(p_1) \times \cdots \times D(p_n)}\), and the solution synthesis \(\sigma_p\) is a function defined on \(\mathbb{R}^{T(p_1) \times \cdots \times T(p_n)} \rightarrow \mathbb{R}^{T(p)}\).

Intuitively, the problem reduction process for an analytics procedure \(p\) is to (1) find \(k\) analytics procedures from \(p_1, \ldots, p_n\), (2) assign a task input \(x_i \in D(p_i)\) for all \(i = 1, \ldots, n\) based on the task input \(x \in D(p)\), and (3) assign a real-valued weight to each assigned task input \(x_i\) based on

---

3 [http://www.wikihow.com](http://www.wikihow.com)
4 [http://www.ehow.com](http://www.ehow.com)
5 [http://www.wikihow.com/Choose-an-Android-Phone](http://www.wikihow.com/Choose-an-Android-Phone)
the real-valued certainty of the original task input $x$. If any analytics procedure $p'$ is not selected in the first step, or an input $x'$ is not assigned to analytics task $p_i$ being selected in the first step, the problem reduction function can simply assign a weight of 0; if any analytics procedure $p'$ is not selected in the first step, or an input $x'$ is not assigned to analytics task $p_i$ being selected in the first step, the problem reduction function can simply assign a weight of 0. Similarly, the solution synthesis process for an analytics procedure $p$ is to (1) find $k$ analytics procedures from $p_1, \ldots, p_n$, (2) assign a task output $y \in T(p)$ based on the task outputs from the impacting analytics procedures $y_i \in T(p_i)$ for all $i = 1, \ldots, n$, and (3) assign a real-valued weight to each task output $y_i$ based on the real-valued certainty of the task outputs $y_i$.

We note that if no weight or certainty is involved in the decision process, i.e. the problem reduction and solution synthesis in the decision process is deterministic, then we can simplify the definition for both processes as $\rho_p : D(p) \rightarrow D(p_1) \times \ldots \times D(p_n)$ and $\sigma_p : T(p_1) \times \ldots \times T(p_n) \rightarrow T(p)$. However, there are a few reasons we might want to consider the certainty and weight. First, some implementation of the analytics procedure can generate more than one conclusions, each associating with some confidence score, which can be utilized by the downstream components to improve the overall performance. Second, users can specify their preference to the analytics procedures that are used in the problem reduction and/or solution synthesis processes by assigning different weight values. Third, these two processes are isomorphic to gateway nodes in the Business Process Model and Notation (BPMN) [51]. While the deterministic versions provide the hard AND-join like aggregation strategy, the generalized versions provide a probabilistic AND connection, which can be used to optimize the execution cost by prioritizing and pruning the branches.

**Definition 4** (Decision process, decision logic, decision factor). An analytics procedure $p$ for analytics tasks $D(p) \rightarrow T(p)$ is a decision process if it contains a set of sub analytics procedures: $p_1, \ldots, p_n$ focusing on individual subtasks $D(p_i) \rightarrow T(p_i)$ for each $i = 1, \ldots, n$ such that there exist a problem reduction $\rho_p$ and a solution synthesis $\sigma_p$ that are defined on $\mathbb{R}^{D(p)} \rightarrow \mathbb{R}^{D(p_1) \times \ldots \times D(p_n)}$ and $\mathbb{R}^{T(p_1) \times \ldots \times T(p_n)} \rightarrow \mathbb{R}^{T(p)}$ respectively. The pair $(\rho_p, \sigma_p)$ is referred to as the decision logic of the decision process $p$. Each sub analytics procedure $p_i$ in the decision process $p$ (with a non-zero weight for some input $x_i$) is called a decision factor of $p$. The set of decision factors $p_1, \ldots, p_n$ is denoted by $\mathcal{F}(p)$.

A complete analytics procedure for a complex analytics task may involve multiple hierarchical procedures combined with linear-chain procedures. For example, we often see that both execution steps and decision steps are interlaced in the how-to manuals. In fact, researchers have also studied a formal taxonomy of business processes based on their complexity, e.g. if it contains cyclic or acyclic processes, if the gateway contains OR operation, etc. [125]. In this thesis, we consider the analytics procedures constructed only from analysis steps, execution processes, and decision processes, and we thus define analytics procedures inductively as follows.

**Definition 5** (Analytics procedure, design process). A design process $\delta : (x, y) \rightarrow p$ aims to find an analytics procedure $p$ for an analytics task $(x, y)$. An analytics procedure $p$ designed for task $(x, y)$ is defined as

1. an analysis step, or
2. an execution process that contains analytics procedures $p_1, \ldots, p_n$, each designed by $\delta$, or
a decision process that contains analytics procedures $p_1, \ldots, p_n$, each designed by $\delta$.

Our definition of analytics procedures is a type of process model, which simplifies the general models such as BPMN [51] or Influence Diagram (ID) or decision network [63], where the latter proposals characterize any arbitrary process as a task-flow graph and thus can represent any structured decision problem under uncertainty in a more flexible way. In addition, our definition gives concrete semantics to each process component (analysis step, execution process and decision process). In fact, based on our definition, analytics procedures are canonical process components [159], i.e. if we represent the analytics procedures using the general graph-based process model, then the subgraph induced by the flows in the analytics procedure has a single entry node and a single exit node (a.k.a. process components), and any two analytics procedures are either disjoint or one is contained in the other (a.k.a canonical). According to the classification of canonical process components [125, 159], analysis steps, execution processes, and decision processes belong to trivial, polygon, and bond components respectively. The analytics procedures are also structured [74], i.e. every node with multiple outgoing arcs (split) there is a corresponding node with multiple incoming arcs (join) and vice versa. In our case, the only split type in analytics procedures is the AND-join like problem reduction $\rho$, and the only join type is the AND-join like solution synthesis $\sigma$. Naturally, any analytics procedure can be represented as a refined process structure tree [159] with the root being the end-to-end analytics procedure and the leaves being the analysis steps.

Once the analytics procedure is determined either manually or automatically through the design process $\gamma$, the developer should create an information system following the analytics procedure. Since the building blocks of all analytics procedures are analysis steps, developers need to develop or integrate an executable module complying the definition of each analysis step. Software development best practices and architecture frameworks, e.g. UIMA [41, 42], GATE [26, 27], have encouraged developers to decouple configuration from logic. As an example, we consider Question Named Entity Recognition (NER), a key analysis step in understanding the semantics from natural language representation, where the input $x$ is a text, and the output $x$ is a list of named entities. One could use a rule-based NER, a CRF-based NER, or a knowledge base lookup based NER. The configuration parameter value could be the set of rules to use for the rule-based NER, a weight vector trained for the CRF model for the CRF-based NER, and the knowledge base to be used by NER. Here, we formally define analysis component, parameter and configurations.

**Definition 6 (Component, parameter, configuration).** An analysis component or component $f$ is a primitive processing unit. Each component is also associated with a set of parameters, denoted by $\{\omega_i\}_i$, which constitute a component configuration $\omega$. If all the parameter values are specified for a component, we call it a configured component $(f, \omega)$.

According to the definition of component and configuration, we can write the output produced by $(f, \omega)$ as $y = f(x; \omega)$, where $x$ is an input (typed object) and $y$ is the output. Note that parameters are not restricted to numeric values, but can hold a reference to any typed object. Once the analysis components and configuration values are determined for all the analysis steps, an information system that can accomplish the original analytics task can be constructed from the analytics procedure. We define an analytics engine or information system and an instantiation
process \( \gamma \) inductively in parallel to the construction of analytics procedure.\(^6\)

**Definition 7** (Instantiation, analytics engine). An **instantiation** process \( \gamma : p \to s \) maps an analytics procedure \( P \) for an analytics task \((x, y)\) to an analytics engine (or information system) \( s \) that processes \( x \) and produces \( y \). An **analytics engine** \( s \) instantiated from \( p \) by \( \gamma \) is defined as

- a \( p \)-compatible configured component \( \langle f, \omega \rangle \), i.e. it is defined on the same domain \( D(p) \to T(p) \) as \( p \), if \( p \) is an analysis step,
- a processing unit that chains in series sub analytics engines \( s_1, \ldots, s_n \), where \( s_i = \gamma(p_i) \), if \( p \) is an execution process that contains \( p_1, \ldots, p_n \).
- a processing unit that aggregates in parallel sub analytics engines \( s_1, \ldots, s_n \) and the decision logic \( (\rho, \sigma) \), where \( s_i = \gamma(p_i) \), if \( p \) is a decision process that contains \( p_1, \ldots, p_n \).

From the definition of analytics engine, we can easily see that an information system is also a process model, more specifically structured process model, like an analytics procedure. The topological structures of both process models are identical given a predefined installation process \( \gamma \). However, in the next section, when we study the analytics meta learning problem, we may vary \( \gamma \) or even the analytics procedure \( p \) to obtain a family of information system for an analytics task.

We use the term **trace** \( s_1 \to \ldots \to s_n \) to describe the processing path of an information system when it is applied to process an analytics task, i.e. a topologically sorted list of all analytics engines, \( s_1, \ldots, s_n \), where \( p_1, \ldots, p_n \) are all the (sub) analytics procedures, in an order such that for any input requirement \( x_i \) of analytics procedure \( p_i \), there exists some procedure \( p_j \) \((j < i)\) that produce \( x_i \) as an output. We call an information system \( s = \gamma(p) \) a **pipeline** if \( p \) is a linear-chain execution process and does not include any decision process. The trace of a pipeline is unique, and can be described by the configured components integrated for each analysis step, e.g. \( \langle f, \omega \rangle_1 \to \ldots \to \langle f, \omega \rangle_n \). We take the typical factoid question answering pipeline again as an example, which is comprised of question analysis, document retrieval, passage extraction, and answer generation four analysis steps in an execution process. A trace would be a unique combination of components, e.g.

```
query tokenization by whitespace string splitter
\to document retrieval from Indri repository index with default parameters
\to sentence extraction based on LingPipe sentence segmenter and Vector Space Model similarity calculator
\to answer generation based on LingPipe NER with a pretrained model and frequency counting of named entities
```

In this section, we review the process of creating an information system for an analytics task by first sketching an analytics procedure via design process \( \delta \) and filling in analysis components for all the analysis steps via instantiation process \( \gamma \), and accordingly we define the terminology along this process.

\(^6\)Although both *analytics engine* and *information system* have the same definition, we use *analytics engine* more often to refer to a subroutine and *information system* to refer to a complete solution for the original task.
2.3 Problem Definition & Solution Framework

In Section 2.2, we introduce the terminology related to developing information systems for analytics tasks. However, as the analytics task becomes more sophisticated and thus more parties are involved, the development process might involve more options and uncertainty. The choice of all the basic elements of an information system – analytics procedures, analysis components, and configurations – may change over time and differ across the stakeholders, other researchers, and the massive ordinary Web users. For example, different directions or depths of analytic thinking may lead to different decision processes with distinct decision factors and/or dependencies even if the objective is the same. As a result, a primitive factor in one decision process can be further decomposed into sub-factors or expanded into an execution process in a more in-depth analytics procedure.

This observation motivates us to extend the processes models defined for a single analytics procedure and a single instantiated information system to describe all the possibilities and variations during the development of information systems, which provides a foundation for analytics meta learning. In the first part of this section, we introduce the concepts of procedure space, system space, and analytics space, and then formally define the problem of analytics meta learning.

**Definition 8 (Procedure space).** For a given analytics task \((x, y)\), the **procedure space** \(P\) is defined as the set of analytics procedures designed for the task via some decision process \(\delta\), i.e.

\[
P = \{ p : x \in D(p) \land y \in T(p) \}.
\]

The procedure space can be considered as a family of process models that have the same start point \((x)\) and end point \((y)\), and any process in the procedure space can individually accomplish the task. Therefore, according to the definition of OR-join gateway, the entire procedure space can also be viewed as a single process model with all the analytics procedure connected by a OR operator.

**Definition 9 (System space).** For a given analytics procedure \(p\), the **system space** \(S\) is defined as the set of information systems instantiated from the analytics procedure via some instantiation process \(\gamma\), i.e.

\[
S = \begin{cases} 
\{ (f, \omega) : x \in D(f) \land y \in T(f) \} & \text{if } p \text{ is an analysis step} \\
S_1 \times \ldots \times S_n & \text{if } p \text{ is an execution process or decision process containing } p_1, \ldots, p_n, \text{where } S_i \text{ is the system space for } p_i
\end{cases}
\]

The inductive definition of system space allows us to proof that the system space can also be represented by the Cartesian product of the analysis components that are used in all the analysis steps of \(p\) given a certain order (e.g. trace), i.e.

\[
S = \{ (\langle f, \omega \rangle_1, \ldots, \langle f, \omega \rangle_n) : f_i \text{ is a } p_i\text{-compatible component and } p_i \text{ is an analysis step in } p \}
\]

Therefore, a system space is also referred to as a **configuration space**. Similar to the procedure space, the system space can also be viewed as a single augmented process model, where each
analysis step in the original procedure $p$ is replaced by an OR-operator connected configured components compatible with $p$.

If we further vary both the design process $\delta$ and the instantiation process $\gamma$, we would obtain the analytics space.

**Definition 10** (Analytics space). For a given analytics task $(x, y)$, the **analytics space** $A$ is defined as the set of analytics engines designed for the task via some decision process $\delta$ and instantiated via some instantiation process $\gamma$, i.e.

$$A = \{ s : x \in D(s) \land y \in T(s) \}.$$  

Again, the analysis space can be viewed as a process model that replaces each analytics procedure in the procedure space model with a system space model. In an analytics space model, the OR gateways are used to connect options between analytics procedures as well as configured components.

In the traditional business process modeling and execution, OR implies equal effect from interchangeable subroutines, and thus no difference can be observed from the outcomes, which however does not hold for an information system. First, runtime performance such as memory usage, CPU time, network bandwidth requirement, etc., might vary across different procedures and components. Therefore, if multiple information systems can produce the same output, the information system that optimizes the runtime performance is always preferred [141]. Moreover, for an analytics task, the stakeholders tend to evaluate the “intelligence” aspect of the information system in term of the task performance, i.e. how the system produced result compares with a known phenomenon or an expert insight, in addition to the runtime performance. In other words, although any combination of an analytics procedure and a set of configured components in the analytics space can be composed as an information system $s$ to find the outcome $y = s(x)$ “in theory”, there is no guarantee that the $y$ is the same or equally optimal in practice.

Analytics meta learning, unlike many other automatic system configuration and optimization solutions, aims to assist system developers in evaluating an entire analytics space, rather than a single system or a system space, and automatically optimizing the analytics for any given task. The key to the problem is to proactively arise awareness of the performance properties (including runtime performance and task performance) at different granularities – from a configuration parameter of an analysis component to the overall information system, so that the right decision can be made at a timely manner at each OR connector. While the runtime performance can be estimated when a component is executed at runtime or even at compile time [141], the task performance can only be judged using an evaluation metric on a gold standard output.

**Definition 11** (Evaluation metric). An **evaluation metric** or **metric** $d$ is a function defined to map $(y, \hat{y}) \in \mathcal{O}^2 \rightarrow \mathbb{R}$, where $y$ is the gold standard output, $\hat{y}$ is the system output, and $\mathcal{O}$ is the output type set. An **aggregated evaluation metric** is defined on a set of outputs in a dataset.

We give some examples of evaluation metrics and aggregated evaluation metrics. If the output can be represented by a short concise word or key phrase, then the evaluation metric can be as simple as a binary scoring function $d(y, \hat{y}) = [y = \hat{y}]$, where 1 represents “agreed”, “satisfied”, “relevant”, etc., and 0 represents “disagreed”, “unsatisfied”, “irrelevant”, or a graded scoring
function, where the higher the grade, the more satisfied or relevant the output is. If the output can be represented by a set of elements, e.g. nuggets in an answer passage \cite{31, 106}, then the classification-based evaluation metrics, such as precision, recall and f-measure, can be calculated. If the output can further be represented by a list of element ranked by importance, e.g. search results, then the ranking-based evaluation metrics, such as average precision, reciprocal rank \cite{161}, can be used. To aggregate the instance-level evaluation measurements, algebraic mean is often used, while sometimes geometric mean is also used \cite{154}.

In this thesis, we use **supervision** for analytics meta learning.

**Problem 1** (Analytics meta learning). Given a set of analytics tasks with task requirements \(X \subseteq \mathcal{I}\) and the corresponding known (expected) outcomes \(Y \subseteq \mathcal{O}\), **analytics meta learning** aims to find the information system \(s^*\) from the analytics space \(A\) that are designed and instantiated for the task \(\mathcal{I} \rightarrow \mathcal{O}\), such that the overall utility is optimized, i.e.

\[
    s^* = \operatorname{argmax}_{s \in A} u(s; X, Y) = \operatorname{argmax}_{s \in A} u_{\text{runtime}}(s; X) + u_{\text{task}}(Y, \hat{Y})
\]  

(2.1)

The utility function \(u(s; X, Y)\) can be combination of runtime performance and task performance. Since the runtime performance does not require a gold standard output, we can safely remove \(Y\) from \(u_{\text{runtime}}\). As the task performance is only measured wherever a gold standard output is available, we can directly apply \(s\) over \(X\) to obtain the system output set \(\hat{Y}\) and compare with the gold standard output set \(Y\), as the second term shown in the last step of Eq. 2.1.

We also consider a more realistic version of the analytics meta learning problem, where we have limited resource to allow any human or machine to learn and discover the optimal information system. Resources used by a component include execution time, storage space, network bandwidth, etc., which can be measured by CPU time, allocated memory size, and data transfers respectively; a resource utilization measure can also be a more specific function of component characteristics (e.g., the cost to execute a configured component on Amazon Web Services\footnote{https://aws.amazon.com/} is a function of execution time and hardware capacity utilized).

**Problem 2** (Analytics meta learning under limited resource). Given a set of analytics task with task requirement \(X \subseteq \mathcal{I}\) and the corresponding known (expected) outcome \(Y \subseteq \mathcal{O}\), **analytics meta learning under limited resource** aims to find the information system \(s^*\) from the analytics space \(A\) that are designed and instantiated for the task \(\mathcal{I} \rightarrow \mathcal{O}\), such that the overall utility is optimized and the processing cost in the learning process is bounded by a resource capacity \(C\), i.e.

\[
    s^* = \operatorname{argmax}_{s \in A'} \mathbf{E}_{x, y} u(s; x, y) \quad \text{s.t. } c(A', X') < C
\]

(2.2)

where we sample a sub analytics space \(A'\) from \(A\), and a sub development set \((X', Y')\) from \((X, Y)\) due to the resource constraint. We use \(c(A, X)\) to measure the overall processing cost of processing paths in \(A\) over the input set \(X\). Since we can have only successfully constructed information systems in the sub space \(A'\), the optimal information that we can obtain \(s^*\) can only belong to \(A'\). In contrast, we intend to ensure the constructed information system \(s^*\) is generalizable to other
unprocessed data in the development set \((X, Y)\), we calculate the expectation over the instance random variable \((x, y)\).

In the second half of this section, we present a general recipe for developing an optimal information system for a complex analytics task using analytics meta learning methodology, which consists of

- **Analytics procedure definition**: defining the analytics procedure, in order to design an architecture/framework that satisfies the task requirement,
- **Analysis component construction**: implementing compatible and configurable lower-level analysis components, and
- **Analytics space exploration**: applying an exploration strategy for automatic pipeline evaluation and iterative planning, which eventually determines the optimal information system over the analytics space under a limited resource assumption.

These three key steps define a development iteration or increment in the entire development life cycle, which can be repeated or paired with traditional development cycles that involve human decision making to complete a development process. In Chapter 3, we study the algorithms that can help users define a procedure space automatically from user created contents. In Chapter 4, we study how to optimize the objective function Eqs. 2.1 and 2.2. Analysis component construction is relatively a task-specific problem, which is often studied in each respective field such NLP, Information Retrieval (IR), etc. From Chapter 6, we empirically validate the proposed methodology using case studies, where we demonstrate how analysis components are constructed.

Finally, we compare analytics meta learning with the traditional predictive analytics using supervised learning. Although both problems assume to learn from either a batch of user feedback (e.g. a training set of input and expected output examples) or interaction with real users, and evaluated using task and/or runtime performance, esp. machine learning community has recently paid much attention to the energy efficiency, existing supervised learning approaches can hardly be directly applied to a complex analytics task. The solution to a complex analytics task is a multi-step information processing system, which can be more conveniently represented, from a system architecture perspective, by a processing trace in the analytics space, whereas the solution space to a single-step prediction task is usually represented by a feature space in the supervised learning setting. We make further comparison between the concepts that have been widely adopted in both problems in Table 2.1.

### 2.4 Analytics Procedure Definition

Based on the prior achievements in the development of configuration space representation (using ECD) and decision process template representation (using DPT), we intent to develop a unified procedure presentation, which allows cross reference between processes and specification of alternative options of analysis components/configurations.

[PROPOSED WORK, C.F. SECTION 2.5 AND CHAPTER 9]
Table 2.1: Comparison between *supervised learning* and *analytics meta learning*

<table>
<thead>
<tr>
<th></th>
<th>Supervised learning</th>
<th>Analytics meta learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target problem</strong></td>
<td>Single-step prediction task</td>
<td>Multi-step complex analytics task</td>
</tr>
<tr>
<td><strong>Supervision</strong></td>
<td>User feedback (e.g. Training set of input/output examples)</td>
<td>Task performance and runtime performance</td>
</tr>
<tr>
<td><strong>Performance measure</strong></td>
<td>Feature vector</td>
<td>Information system (or flattened processing trace)</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>Feature space</td>
<td>Analytics space</td>
</tr>
<tr>
<td><strong>Operation space</strong></td>
<td>Feature engineering</td>
<td>Procedure definition and component construction</td>
</tr>
<tr>
<td><strong>Instance preparation</strong></td>
<td>Parameter estimation</td>
<td>Analytics space exploration</td>
</tr>
<tr>
<td><strong>Optimization process</strong></td>
<td>Search algorithms</td>
<td>Exploration strategies</td>
</tr>
<tr>
<td><strong>Optimization schemes</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.5 Proposed Work

We intend to develop a unified procedure presentation, which allows cross reference between processes (similar to DPT) and specification of alternative options (similar to ECD).
Chapter 3

Procedural Knowledge Discovery

Despite manual specification of analytics procedures as described in Chapter 2, this chapter focuses on a subfield of analytics meta learning – automatic procedure discovery algorithms. We explore a wide range of human-generated data sources, such as community created and maintained semi-structured procedural knowledge base, search query log, as well as procedural knowledge embedded text, and design algorithms to mine such knowledge and canonicalize the representation from each respective source.

In Section 3.1, we first review prior work related to procedural knowledge extraction, which include extraction from semi-structured procedural text, script extraction, and causal relation and argumentation extraction. We then in Section 3.2 give our definition of procedural knowledge base and introduce the features that can be extracted from semi-structured procedural knowledge bases, which is utilized in the following sections for training purpose. Before we study the algorithms to expand a procedural knowledge base or construct one from scratch, we demonstrate how to leverage an existing procedural knowledge base for information retrieval applications. We take search task suggestion or query suggestion as an example in Section 3.3.

We then propose to mine procedural knowledge from various types of data sources. Section 3.4 describes a supervised learning approach to take advantage of search activities to identify task procedures from search query log, and presents the experiment results using the proposed approach. Sections 3.5 and 3.6 then focus on extracting such knowledge from unstructured text and community QA site. Finally, in Section 3.7 we propose future work to explore additional sources, such as on-line community questions answering sites, bibliographic databases, etc.

3.1 Related Work

In this section, we review prior work related to procedural knowledge extraction, which include extraction from semi-structured procedural text, script extraction, and causal relation and argumentation extraction.
3.1.1 Instruction Extraction from Semi-Structured Text

Several automatic procedural knowledge base construction approaches have also been proposed to extract instructions from semi-structured texts, e.g. eHow or wikiHow articles [2, 70, 116, 121], recipes [121], community-based QA [9], etc. Most approaches take advantage of structural information (e.g. HTML tags [9], enumeration indicators [34, 70]), and define rules or templates to extract textual content. In a separate step, NLP tools are applied to extract relations and normalize each goal and action to its ontological form, to support linking to other ontological resources. Researchers have also studied how to identify script knowledge [137] which focuses on the temporal ordering of events [19]. People [2, 70, 116, 121] have also attempted to construct structured procedural knowledge bases by defining ontologies or incorporating other existing representations such as PDDL [45].

In contrast, our approach takes advantage of the writing style of semi-structured procedural texts and proposes a set of structural and textual features for identifying procedural knowledge; the implemented approach can be optimized with a supervised learning method. Moreover, beyond the conventional use of small-scale procedural knowledge in AI planning [110] or NLP [9], our thesis also studies the problem of how to apply a large-scale procedural knowledge base to complex task search problems.

3.1.2 Script Text Extraction

Script text extraction is also a term in the NLP community referring to extracting event sequence from text, which is related to execution process discovery. We propose to further explore the latest achievement in script text extraction.

3.1.3 Causal Relation & Argumentation Extraction

Causal relation and argumentation are often used to describe decision logic. A number of features and models have been proposed to extract two rhetorical structures relevant to our problem: causal and argumentative relations, in fact, identifying both structures would play a major role in decision making support [142, 145]. However, little research has attempted to formally study to directly identify a set of decision factors for a decision goal in order to construct decision processes for decision support.

Causal relations are usually identified from formal documents such as reports [142], scientific articles [103], and social problem related Web pages [56], whereas argumentation is identified from persuasive essays [145] or debate texts [55]. Such relations are identified by classifiers trained on lexical features [15, 20], using PDTB corpus [130], or domain corpora [104, 126]. A number of discourse parsers [86, 124] have also been developed.

We further propose to explore the tools, algorithms, and datasets that have been developed recently.
3.2 Procedural Knowledge Base

Many knowledge bases such as Wikipedia or Wikidata that have been widely utilized contain a huge amount of descriptive knowledge. Procedural knowledge [6, 47], also known as know-how, is the knowledge exercised in the accomplishment of a task, i.e., how to do things, which is usually acquired by experience and considered tacit and not easily shared, compared to descriptive knowledge. However, shared explicit procedural knowledge lays a foundation for efficiently coordinated action, which is often referred to as best practices or business rules within communities or organizations. As wikiHow and many similar how-to Web sites allow users to easily share procedural knowledge, semi-structured large-scale procedural knowledge bases have become available.

In this section, we give our definition to procedural knowledge base. In Sections 2.2 and 2.3, we have given definitions to a single procedure and a family of procedures that achieve the same goal. In this section, we consider the collection of all procedures that involve tasks, which we refer to as procedural knowledge base. Procedural knowledge captures how people decompose a complex task into a mixture of several mental actions (e.g., “choose”, “determine”, etc.) and/or physical actions (e.g., “visit”, “store”, etc.), which correspond to the decision process and execution process.

**Definition 12** (Is-achieved-by relation, procedural knowledge base). We refer to the relation between the task and the subtasks in the execution and decision process as is-achieved-by relation, which connects all the task descriptions. **Procedural knowledge base** (or procedural knowledge graph) $G = (T, R)$ is the set of all tasks $T$ and all is-achieved-by relations $R$. A weight $w(t, s)$ may also associate with each relation $r(t, s) \in R$.

We can see from the definition that an analytics procedure $p$ can be considered as a tree induced from $G$, with the root being original task.

3.2.1 Semi-Structured Procedural Knowledge Base

A widely used type of procedural knowledge base is semi-structured procedural knowledge base. We extend the definition of task in Section 2.2 to consider the semi-structured representation, which includes both the task description and a textual description about the procedure, as we will see that even in a semi-structured procedural knowledge base, the task requirement, objective and action are stated usually in a single paragraph.

**Definition 13** (Task description in semi-structured procedural knowledge base). A task $t$ is represented by a short and concise summary and a detailed explanation for clarification in semi-structured procedural knowledge base.

A semi-structured procedural knowledge base such as wikiHow has an identical structure, but it allows users to more easily view and edit; each article corresponds to a task, whose summary is given in the title and whose explanation is detailed in the introduction section. Each step (or substep) also corresponds to a task, whose summary is given in a bold text at the beginning of the step, which is followed by the explanation. The is-achieved-by relations have three representation forms: numbered steps in each article page (first-level subtasks), bulleted substeps under each
step description (second-level subtasks), and free links[^1] directed from a step to another task page. Steps that have no substeps or outgoing free links are considered primitive tasks. Current semi-structured procedural knowledge bases do not allow editors to explicitly specify the relation strength between tasks, or the importance of each subtask[^2] Therefore, we assume each relation shares an equal weight with all other outgoing relations, i.e. \( w(t, s) = 1/\text{od}(t) \), where \( \text{od} \) is the out-degree of the task \( t \).

We attempt to leverage the style guidelines for writing semi-structured procedural knowledge, such as the wikiHow guide[^3], which indicates that an action-oriented instruction beginning with a verb is required at the beginning of each procedural step, but we also attempt to process articles that do not fully comply with the guide. We accordingly propose a set of textual features and structural features to identify query spans from each task description, and then adapt similar features to extract wikiHow-style procedural knowledge descriptions from search queries and relevant text snippets.

**Location (LOC).** As suggested in the wikiHow writing guide, a task should provide both “skimmable information that readers can quickly understand” in the title of the article and in the beginning sentence of each step, and “rich informative articles for more patient users” in the article’s introduction and in the sentences which follow in the detailed explanation of each step. Therefore, we define features to capture the location of each word.

**Sentiment of event mention (SEN).** Similar to other wiki-based knowledge bases such as Wikipedia, wikiHow discourages personal or subjective reference in the description (such as use of “I”, “we”, etc.), which motivates us to also detect sentence-level sentiment features such as polarity, subjectivity, modality (certainty degree) and factuality[^135], and assign the same feature value to all the tokens in the sentence.

**Part of speech (POS).** Since both the article title and the first sentence in each step are required to begin with a verb in bare infinitive form, we also extract the fine-grained part-of-speech value for each token.

**Parsing (PAR).** To further understand the task facets such as occurrences of subsidiary resources (e.g. a target object) or constraints (e.g. a duration), we include features extracted from dependency parsing, named entity extraction, and chunking.

**Word and context.** As in other natural language sequence labeling tasks, we also define features including the token’s surface form, its stem form, its TF-IDF score, as well as the word features and the part-of-speech tags of the previous and the next words in the task summary or explanation.

A complete list of features is presented in Table [3.1](http://en.wikipedia.org/wiki/Help:Wiki_markup#Free_links)

[^2]: As users may verbally express their confidence and/or their attitude toward the importance in the task explanation section, we can hence incorporate richer linguistic features such as modality particles, verbal auxiliaries, etc. to identify verbal expression of weight.
Table 3.1: Semi-structured procedural knowledge base feature definition

<table>
<thead>
<tr>
<th>Description</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $w_i$ appears in the task summary and explanation</td>
<td>2 or 0</td>
</tr>
<tr>
<td>The sentence that contains $w_i$ has a positive, negative, and neutral stance</td>
<td>3</td>
</tr>
<tr>
<td>The sentence that contains $w_i$ is subjective and objective</td>
<td>2</td>
</tr>
<tr>
<td>Whether $w_i$’s POS is one of the modality types used in Slinket</td>
<td>2</td>
</tr>
<tr>
<td>Fine-grained POS tags of $w_i$ used in the Penn Treebank</td>
<td>36</td>
</tr>
<tr>
<td>Basic Stanford dependency types of $w_i$</td>
<td>50</td>
</tr>
<tr>
<td>If $w_i$ is inside a named entity, noun phrase and verb phrase</td>
<td>3</td>
</tr>
<tr>
<td>Surface form, stem, and TF-IDF score of $w_i$</td>
<td>3</td>
</tr>
<tr>
<td>Surface form, stem, TF-IDF score and POS tags of $w_{i-1}$</td>
<td>39</td>
</tr>
<tr>
<td>Surface form, stem, TF-IDF score and POS tags of $w_{i+1}$</td>
<td>39</td>
</tr>
</tbody>
</table>

3.3 Application: Search Task Suggestion

Before we further discuss how to extract procedural knowledge from various sources, we demonstrate how to leverage existing procedural knowledge for information retrieval related applications beyond analytics meta learning. In this section, we consider a widely studied topic in information retrieval – search task suggestion or specifically query suggestion in the context of search engine. We focus on task-oriented search tasks, where the objective is to accomplish real-life tasks. For example, to organize a conference (task), the organizer needs to look for information related to choosing a hotel, identifying and deciding among banquet options, recruiting volunteers, selecting and contacting publishers, etc (subtasks). For each specific task, users must gather more detailed information. For example, when choosing a hotel, the organizer must consider the number and size of conference rooms, potential arrangements for meal catering and menu planning, whether or not a discounted rate will be available, etc. There is a huge demand for search engines to better assist their users in achieving their intended goals for such ad hoc tasks. We call the problem task-oriented search task suggestion.

Researchers have studied problems related to search intent analysis for general queries; for example, how to identify search intents [18, 23, 28, 53, 73, 82, 89, 128], how to suggest search queries to the user [7, 25, 69, 99, 164], how to rank results that cover diverse aspects of the task, etc. Most studies have so far mostly relied on queries and search logs, search result texts, behavioral information, etc. to improve search quality for entity-centric queries [30]. Recently, research has shown that structured knowledge bases (such as Wikidata or formerly FreeBase) or semi-structured knowledge bases (such as Wikipedia) can be used to improve search quality and experience [16, 29, 84, 172].

When users turn to search engines for information seeking and problem solving, we investigate

4(Sub)task in this thesis refers to an action that is performed to achieve a goal in a general context [91], as we defined in Section 2.2, which differs from the notion of task in the search scenario [57, 58]. We use (sub) search task to denote the latter case.

5http://wikidata.org/
6http://wikipedia.org/
how existing procedural knowledge can be leveraged to help understand a user’s search intent and suggest sub search tasks to users.

**Problem 3** (Task-oriented search task suggestion, STS). Given a procedural knowledge graph $G$ and a task-oriented search task $q_0$, we aim to (a) identify the task $t_0 \in T$ the user intends to accomplish, and then (b) retrieve a list of $k$ subtasks $t_1, \ldots, t_k \in T$ and also (c) suggest the corresponding sub search tasks $q_1, \ldots, q_k$.

A search task is usually represented by a search query or a key phrase [57, 58, 91]. Similarly, we assume both the input search task $q_0$ and the suggested search tasks $q_1, \ldots, q_k$ are queryable phrases. As a result, the problem can be straightforwardly solved by suggesting relevant (sub)queries for task-oriented queries. In contrast to the entities and attributes in a descriptive knowledge base, which are usually words or phrases and can used directly in query suggestion, summaries and explanations in a procedural knowledge usually contain richer action and condition information. Therefore, the problem poses three challenges. First, we need to identify the task $t_0 \in T$ the user intends to accomplish. If the query corresponding to the search task exactly matches the summary of a task $t_0$ (exact string match or after lexical transformations), then we can directly return $t_0$. However, in most cases when an exact match does not exist, we need to incorporate additional information such as the search result page of the query and the full description of each task, and rank candidate tasks using a retrieval method. Second, we need to choose the tasks $t_1, \ldots, t_k \in T$ to suggest and then prioritize them based on the knowledge $G$ containing relations $V$. Intuitively, we can choose the task $t_0$’s top-$k$ subtasks ordered by the weight of the is-achieved-by relation, but we may consider second-level subtasks (e.g. the bulleted items in wikiHow) as well. Finally, we need to extract queryable phrases from each task description, and the algorithm should thus learn how searchers have been trained to formulate queries.

### 3.3.1 Modeling & Solution

In this section, we propose a supervised approach to learn how searchers have been trained to formulate task-oriented queries from search query log, and applied the model on the semi-structured procedural knowledge base to extract queryable phrases. The challenge here is to identify only the task-oriented queries (instead of entity-centric queries) from a list of candidate queries to annotate the task descriptions. We create a collection of task descriptions with task-oriented query phrases annotated in an iterative manner.

We first identify the exact matching pairs of searchers’ issued search tasks and task descriptions in the procedural knowledge base, although we expect that many lexically distinct but semantically identical pairs may exist that will not be extracted. To achieve this, we may scan through the entire search query log to find each query $q$ that matches the description of the task $t$. We add the annotated task description to the corpus. In addition, we collect related queries by combining the user-issued queries from the same session and the list of queries suggested by the search engine for the next iteration. If we have no large query log, we may also manually create task-oriented search tasks for the tasks in the procedural knowledge base. Specifically, we use the summary of the task $t$ to form a search query $q$. Since these search queries are “artificially” created, and do not necessarily represent how searchers normally formulate queries for these tasks in similar
situations, we exclude it from the parallel corpus, which is only used for search engines to suggest real queries for the next iteration.

Once we collect a list of related queries for \( q \), we compare them with the known subtasks of \( t \) in the procedural knowledge base. For each related query \( q' \), we first find the subtasks \( t_1, \ldots, t_n \) that contain \( q' \) in the task description. We then add all the annotated subtasks \( t_1, \ldots, t_n \) to the corpus. As we assume that all necessary subtasks have been detailed in the procedural knowledge base, we consider a related query irrelevant to accomplishing the task if it is not mentioned in any of the subtasks, and it is excluded in the parallel corpus and discarded for future iterations. We also ignore subtasks that are not matched to any related query, a situation that may arise from a lack of evidence from query logs, or truncation due to limited space for displaying suggested queries on a result page. We continue the iterative process.

We use the annotated corpus to train supervised models that can automatically create task-oriented queries for tasks. We view Problem 3 as a word sequence extraction problem, which is usually reduced to a sequence labeling task. For training a sequence labeling model, we employ the \( \text{BIO} \) encoding, i.e., each token in the task description that begins each occurrence of the query phrase is labeled as \( \text{B} \), the tokens that continue in the query phrase are labeled as \( \text{I} \), and all other tokens in the task are labeled as \( \text{O} \). We use the features defined in Section 3.2.1 and then apply a common statistical model for sequence labeling (such as Conditional Random Fields (CRF)), which also enables us to process articles that do not fully comply with a style guide. We train a query construction model \( M_Q \) using the training sets \( (X^t, Y^t) \). To construct search queries for an unseen task \( t \), we first extract the features \( x^t \) from the task description, and then apply the query construction model \( M_Q \) to extract each search query candidate \( q_i \), i.e. to identify each word sequence \( w_i = w_{i1} \ldots w_{il} \) from \( t \) with the corresponding \( y_{ij} \) labeled as \( \text{BIO} \ldots \text{I} \), where \( l_i \) represents the length of \( w_i \), and \( y_{ij}^* \) is the optimal label sequence, i.e.,

\[
y_{ij}^* = \arg\max_{y_i \in \{\text{B},\text{I},\text{O}\}^{|l_i|}} p(y_i^t|x^t_i; M_Q)
\]

where \(|t|\) represents the length of the task \( t \)'s description.

Finally, we present our proposed approach for search task suggestion. Given a task-oriented search task represented by query \( q \), we first retrieve a list of candidate tasks from the procedural knowledge base that mention the query \( q \) in either the summary or the explanation. When more than one task is returned from the procedural knowledge base, we need to determine which task is the best fit for the user’s search intent. Therefore, we leverage the query construction model \( M_Q \) to estimate the likelihood of the \( j \)-th occurrence of the query \( q \) in a retrieved task \( t_i \)'s description (i.e. a word sequence \( w_{ij} = w_{ij1} \ldots w_{ijl} \), where \( l \) is the length of query \( q \)). We select the \( i^*\)-th task that contains the \( j^*\)-th word sequence which maximizes the conditional probability, i.e.

\[
(i^*, j^*) = \arg\max_{i,j} p(y_{ij}^{t_i}|x^t_i; M_Q)
\]

Once we have identified the task \( t \), we identify the tasks in the procedural knowledge base for future query suggestion using a few heuristic strategies. First, we may select only the first-level subtasks and order the weight of the is-achieved-by relation in descending order. We may also select both the first-level and second-level subtasks and use the weight of each first-level subtask
or the weight of each second-level subtask multiplied by the weight of its corresponding first-level subtask.

When a list of subtask candidates \( t_1, \ldots, t_n \) are retrieved from the existing procedural knowledge base, we apply the query construction model \( M^Q \) again, for each subtask \( t_i \)’s summary and explanation to identify each word sequence \( w_{ij} \) labeled as \( \text{BQ}_I \ldots \text{IQ} \) using Eq. 3.1. Among the extracted query candidates \( \{w_{ij}\}_j \) for each subtask \( t_i \), we choose the query \( q_i \) that maximizes Eq. 3.2. The “winning” query candidates \( q_1, \ldots, q_n \) corresponding to the subtask candidates are globally ranked by multiplying the weight obtained from subtask retrieval with the likelihood estimation; finally, we select the top-\( k \) queries. This process guarantees that the queries are extracted from distinct subtask candidates, which can lead to the most coverage of all necessary subtasks. Alternatively, a diversity-aware ranking approach can also be applied to global rank all query candidates \( \{w_{ij}\}_j \).

### 3.3.2 Data Preparation

We used two publicly available data sets: an English wikiHow data dump and the AOL search log and public search engines (Google and Bing) to collect the suggested queries. The data and code used in this section are available to download.

**English wikiHow data dump.** We crawled an English wikiHow data dump using a modified version of the WikiTeam tool which contains 198,163 non-redirect articles within namespace “0” (Main). We also filtered out the articles that are marked as stub (incomplete and need more information) or have no “Introduction” or “Steps”, which results in a collection of 149,975 valid articles. We performed minimal pre-processing to reformat the articles to the MediaWiki format, without changing any textual content.

In particular, we first extracted the top-level task summary and explanation after identifying the title and the introduction of each article. We then located the “Steps” section and extracted the enumerated or bulleted items to build a local subtask hierarchy. Next, we built a procedural knowledge graph by creating nodes representing all the top-level tasks and their subtasks and establishing relations based on both the task-subtask relation as well as internal links. Applying this approach, the constructed procedural knowledge graph contains a total of 1,488,587 tasks and 1,439,217 relations, where 265,509 of the tasks are non-primitive, and 100,605 relations come from internal links. We built a Lucene index for all task descriptions, which supports retrieval of candidate tasks for the search task suggestion problem described in Section 3.3.1.

**AOL search query log.** We also used the publicly available AOL search query log, which consists of over 21M Web queries (over 10M unique queries) collected over three months in 2006 from AOL. We downcased the query strings and task summaries, and removed all
non-alphanumeric characters. We identified that 867 unique queries (corresponding to 9,847 new queries from 23,099 lines) match some task summary in our constructed procedural knowledge base. In the experiment, we identified 639 unique queries that have at least two tokens (corresponding to 3,086 new queries from 7,019 lines), which tend to be less ambiguous and more likely task-oriented. We could estimate that each task-oriented search task is repeated 4.8 times on average, compared to 2.1 times for all queries, supporting the intuition that common task-oriented searches tasks are more often encountered than general search tasks, providing further motivation for our study. The 639 unique queries correspond to 619 tasks if punctuation marks and whitespaces are ignored.

To retrieve the related queries issued by users in the same session from the query log, we first identified related query candidates by collecting the queries that were issued by the same user within 30 minutes after they issued each matching query. In this way, we collected 33,548 query candidates (31,955 unique queries, 50 per query). We did not use other commonly adopted session detection heuristics such as edit distance or cosine similarity between query pairs, since a task-oriented search query may not share words in common with related search queries. For example, we identified a case where “visit niagara falls” is followed by another query “map of northeast” in the AOL search log, “design a living room” is followed by another query “choosing colors”, etc. We instead counted on the annotated corpus construction process (described in Section 3.3.1) to correctly identify which of the query candidates are relevant to accomplishing the task.

**Queries suggested by search engines.** To create “artificial” search tasks, we randomly sampled 1,000 non-primitive tasks from the constructed procedural knowledge that do not appear in the query log. We limited the total number of tasks in building the parallel corpus partly due to our limited access to commercial search engines, and also because in a preliminary experiment we found that performance of the sequence labeling model on the test set had become stable. We merged them with the 639 identified queries from AOL search query log, using commercial search engines (Google and Bing) to build the parallel corpus, which correspond to 1,619 tasks. We constructed each query without using additional operators such as quotes, plus signs, minus sign, etc., unless quotes appear in the summary. Google may suggest up to eight queries at the bottom of the search page for each search task and Bing may suggest up to sixteen queries within the search result column and on the right side of the results page. We collected a total of 9,906 queries suggested by Google (6.11 per query on average, 8 maximum) and 9,715 suggested queries by Bing (5.99 per query on average, 13 maximum).

We notice that both Google and Bing are able to display procedures directly on the result page from wikiHow (only Bing) or other procedural data sources (Google), which allows users to grasp the knowledge without an extra click, but hardly helps users or automatic decision support systems identify how to explore more related information from the Web. We also found that both commercial search engines tend to suggest and display related queries for short queries, as they are more likely topical broad and semantically ambiguous. However, for task-oriented search, the complexity of describing the task does not necessarily reflect the difficulty of accomplishing the task, in terms of which aspects to consider and the steps to perform.

---

12 Also on wikiHow: [http://www.wikihow.com/Visit-Niagara-Falls](http://www.wikihow.com/Visit-Niagara-Falls)
Table 3.2: Averaged number and percentage of related queries suggested by Google or Bing or issued by the users subsequently in the same session that are mentioned by a task description in wikiHow

<table>
<thead>
<tr>
<th></th>
<th>Averaged number</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Google Bing Log</td>
<td>Google Bing Log</td>
</tr>
<tr>
<td>Full phrase</td>
<td>0.070 0.046 0.050</td>
<td>1.162 0.727 1.131</td>
</tr>
<tr>
<td>New words</td>
<td>0.473 0.592 0.164</td>
<td>8.487 9.522 4.639</td>
</tr>
</tbody>
</table>

Annotated corpus. Including the 639 queries that have already been associated with some tasks in the procedural knowledge base, we identified that a total of 1,182 query-task description pairs (from 1,146 distinct descriptions) among the suggested queries can be found in the subtask descriptions of the task corresponding to the original query.

To further understand how many of the related queries are aligned to a task description, we show in Table 3.2 the averaged number and percentage of related queries suggested by Google or Bing or issued by users subsequently in the same session that are mentioned within a task description. “New words” refer to the subset of words that are in the suggested queries but not in the original queries. First, we can see that the queries identified in the same session do not have a higher quality (1.131% – 4.639%) in related task suggestion, which may be due to an oversimplified session detection algorithm. In comparison, related queries collected from commercial search engines tend to be more relevant (0.727% – 9.522%). Moreover, if we focus on the new words, we identify many more (from 0.727% – 1.162% to 8.487% – 9.522%) tasks that are aligned with the queries suggested by the search engines.

We note that if a query cannot be aligned with a task description using our proposed construction method, this does not guarantee that the query is not semantically identical (e.g. a paraphrase) of any task description. In fact, as motivated in Section 3.3.1, the surface form matching approach can simplify the parallel corpus construction process while still guarantee the parallelity.

3.3.3 Experiment Settings

We describe the experiment settings, including the tools we used to extract features and learn the sequence labeling models, the evaluation methods, and baseline methods.

Evaluation methods. We conducted two types of evaluation for both problems. We first evaluate the performance of proposed sequence labeling approach on the query construction task using the annotated corpus. Then, based on manual judgment, we evaluate our proposed end-to-end solution and compare with commercial search engine suggestions.

We performed 10-fold cross validation on the constructed parallel corpus, and report average performance results. We extracted text spans from the predicted labels and used precision, recall and F-1, averaged over the performance on all test instances (macro-averaged) and averaged on each task then across all tasks (micro-averaged), to compare the proposed approach with baseline methods. We also employed two ROUGE scores (F-1 based ROUGE-2 and ROUGE-S4\(^\text{14}\)), after

\(^{14}\)We did not use any unigram based metric (such as ROUGE-1 or ROUGE-SU) or recall based metric because we
we removed common English stop words\footnote{We used /resources/stopwords-rouge-default.txt from ROUGE 2.0 tool. \url{http://kavita-ganesan.com/content/rouge-2.0}} and constructed an array of words in lemma forms.

In the end-to-end comparison, we randomly sampled 50 triples from the corpus and we applied both our approach for search task suggestion and commercial search engine services to produce up to eight related queries. We manually judged whether each search query can obtain a search result that can help achieve the user’s goal. Given that this judgment is binary, we report the macro-averaged and micro-averaged Precision@8, and MAP averaged over all 50 test instances.

In the NTCIR-11 IMine task\cite{91}, importance judgment was conducted manually by the annotators for each subtopic/subtask to obtain a ranked list of gold-standard tasks for a given topic/task, which allowed the use of nDCG and D#-measures\cite{134} for performance evaluation. However, since semi-structured procedural knowledge bases such as wikiHow or eHow and proposed situation ontologies\cite{9, 34, 70, 116} do not explicitly specify an importance rating for each subtask, we consider the subtasks to be unordered.

**Baseline methods.** For the query construction task, we compare the proposed approach CRF with other supervised and unsupervised approaches. We varied the classifier to evaluate HMM, a Hidden Markov Model based sequence labeling approach based on surface forms, LR, a logistic regression classifier, and an SVM classifier. The latter two are trained on the same set of features to classify B, I, and O without the sequence labeling assumption. We also compared performance with models that use only one category of the proposed features (W), all but one category (W/O), and only local or context features. Finally, we compare with a key word extraction method based on TF-IDF (TFIDF), where we try to maximize the macro-averaged F-1 score by tuning the TF-IDF score threshold when determining whether a word is selected as a key word.

**Feature extractors and learners.** We used Stanford CoreNLP\url{http://nlp.stanford.edu/software/corenlp.shtml} Ver. 3.5.2 to extract sentences, tokens, stems, POS tags, dependency labels, chunks (noun phrases and verb phrases), and named entities using the default configuration. We used MALLET\url{http://mallet.cs.umass.edu/} Ver. 2.0.7\cite{98} to learn and test the sequence labeling models (CRF and HMM) with sparse weighting method, LibLinear\url{http://www.csie.ntu.edu.tw/~cjlin/liblinear/} Ver. 1.8 for training and testing LR and SVM models. As the Web documents tend to be noisy, we preprocessed the texts by inserting periods at certain places to improve the parsing performance. Details can be found in the source code.

### 3.3.4 Experimental Results

We first compare the query construction result between the proposed approach and the baseline methods in Table\ref{tab:3.3}. We also conducted a t-test to calculate the significance level of each baseline method compared against the proposed approach (CRF). We can see the proposed CRF-based sequence labeling approach can significantly outperform other baseline classifiers (at a significance level of 0.05 in terms of F-1 and 0.1 in terms of ROUGE).

The performance gap between the sequence labeling approaches and the independent classification-did not set a limit for either the number of chunks or the total length of the text returned for each task.
Table 3.3: Search task suggestion results. A dagger (†) and one to three stars (⋆) represent significance levels of $p \leq 0.1, 0.05, 0.01,$ and 0.001 respectively, and ns represents not significant.

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Ma P</th>
<th>Ma R</th>
<th>Ma F1</th>
<th>Mi P</th>
<th>Mi R</th>
<th>Mi F1</th>
<th>R-2</th>
<th>R-S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF</td>
<td>.8691</td>
<td>.6563</td>
<td>.7471</td>
<td>.9359</td>
<td></td>
<td>.6934</td>
<td>.6930</td>
<td>.8112</td>
</tr>
<tr>
<td>HMM</td>
<td>.5281</td>
<td>.1984</td>
<td>.2870</td>
<td>.9160</td>
<td></td>
<td>.3113</td>
<td>.3161</td>
<td>.5200</td>
</tr>
<tr>
<td>SVM</td>
<td>.7589</td>
<td>⋆⋆⋆</td>
<td>.6248†</td>
<td>.6955</td>
<td>⋆⋆</td>
<td>.8641</td>
<td>.6674</td>
<td>.6612</td>
</tr>
<tr>
<td>LR</td>
<td>.8461</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6144</td>
<td>.7476</td>
<td></td>
</tr>
<tr>
<td>TFIDF</td>
<td>.0087</td>
<td>⋆⋆</td>
<td></td>
<td></td>
<td></td>
<td>.0072</td>
<td>.0079</td>
<td></td>
</tr>
<tr>
<td>POS</td>
<td>.8346</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6015</td>
<td>.9248</td>
<td></td>
</tr>
<tr>
<td>PAR</td>
<td>.6846</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6376</td>
<td>.8811</td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>.2476</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.5980</td>
<td>.1671</td>
<td></td>
</tr>
<tr>
<td>WORD</td>
<td>.8164</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6803</td>
<td>.9105</td>
<td></td>
</tr>
<tr>
<td>W/O subsets of features (±context)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS</td>
<td>.8811†</td>
<td>.4720</td>
<td>.6015</td>
<td>.9248</td>
<td></td>
<td>.5487</td>
<td>.5541</td>
<td>.7109</td>
</tr>
<tr>
<td>PAR</td>
<td>.6846</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6376</td>
<td>.8811</td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>.2476</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.5980</td>
<td>.1671</td>
<td></td>
</tr>
<tr>
<td>WORD</td>
<td>.8164</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6803</td>
<td>.9105</td>
<td></td>
</tr>
<tr>
<td>Local</td>
<td>.7808</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.6737</td>
<td>.8935</td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td>.8155</td>
<td>⋆</td>
<td></td>
<td></td>
<td></td>
<td>.7068</td>
<td>.8933</td>
<td></td>
</tr>
</tbody>
</table>

Based approaches such as LR and SVM suggests that the query construction problem is similar to other sequence labeling problems such as named entity detection or supervised keyword extraction. With only surface form features, HMM performs worse than CRF. All the supervised approaches, which take advantage of the constructed parallel corpus, can outperform the unsupervised approach (TF-IDF).

When we test each feature category (POS, Parsing, Location, Word, Local, and Context) independently, we can see none of them can beat the proposed approach with a confidence level of $\geq 0.99$. We also see that word features contribute the most to performance, which can achieve around 90% of the performance in terms of F-1 and 95% in terms of ROUGE when all the features are used. Both F-1 and ROUGE scores drop when we remove any feature category from the feature list, and they drop the most when Word features are removed, which implies that all the features have positively contributed to the query construction task.

To better understand what non-Word features most likely promote a word to become a B or I, we shows the top-5 features for $O \rightarrow I$, $B \rightarrow I$, $I \rightarrow I$ transitions in Table 3.4. We see that the whole query phrase is very likely extracted from the summary part of a description (sum) due to its clarity and conciseness. We also see that both singular nouns (NN or NNP) and verbs in either VB or VBP forms are selected as indicators for beginning a query, and verb phase (VP) is a useful feature when deciding whether to continue a query.

We then evaluated the end-to-end performance of the proposed STS solution using the query...
Table 3.4: The non-Word features that contribute the most to each label in the search task suggestion problem. “+1”/“-1” presents the feature of the next/previous word, “sum” refers to summary, “VP” represents if the word is inside a verb.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>O → B</td>
<td>POS: NNP</td>
<td>LOC: sum</td>
<td>DEP: ccomp</td>
<td>POS: VB</td>
<td>DEP: nsubjpass</td>
</tr>
<tr>
<td>B → I</td>
<td>POS: VB</td>
<td>LOC: sum</td>
<td>POS: VBP</td>
<td>POS: NNP</td>
<td>POS: NN</td>
</tr>
<tr>
<td>I → I</td>
<td>LOC: sum</td>
<td>POS: IN</td>
<td>VP</td>
<td>DEP: dobj</td>
<td>POS: JJ</td>
</tr>
</tbody>
</table>

Table 3.5: Search task suggestion results

<table>
<thead>
<tr>
<th></th>
<th>PROPOSED</th>
<th>GOOGLE</th>
<th>BING</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro P</td>
<td>.4457</td>
<td>.0972</td>
<td>.0333</td>
<td>.0676</td>
</tr>
<tr>
<td>Micro P</td>
<td>.4457</td>
<td>.0973</td>
<td>.0313</td>
<td>.0612</td>
</tr>
<tr>
<td>MAP</td>
<td>.3361</td>
<td>.0553</td>
<td>.0120</td>
<td>.0549</td>
</tr>
</tbody>
</table>

construction model. We present the results in Table 3.5 and show examples in Table 3.6. We see the proposed solution can outperform the commercial search engines by +0.35 to +0.41 in terms of Precision@8 and +0.28 to +0.32 in terms of MAP. We see that our proposed method, which is tailored for task-oriented search, can provide unique insights when suggesting tasks, compared to current general-purpose commercial search engines, which have been designed for entity-centric search and tend to suggest queries by appending keywords such as “product”, “image”, “logo”, “online”, “free”, etc. We discovered three types of errors from our system: (1) Some suggested queries are ambiguous when presented by themselves (e.g. “fix it”, “install”). (2) duplicated queries are suggested from different subtasks. (3) the manually created knowledge sometimes still contains a few non-instructional “steps”. We could further improve the proposed approach by conducting a co-reference analysis or other approaches to incorporate original queries and contexts in the suggested queries. We could also collectively rank the candidates to avoid suggesting duplicated tasks.

3.4 Procedural Knowledge Extraction from Search Activities

Although community-centric procedural knowledge bases may have collected hundreds of thousands of task descriptions, users still face ad hoc situations (tasks) that are not covered by an existing procedural knowledge base. From this section, we focus on procedural knowledge extraction, which is also referred to as automatic procedural knowledge base construction.

In this section, we hypothesize that other searchers may have faced similar situations in the past and have already interacted with search engines to attempt a solution, which means we may discover implicitly expressed procedural knowledge from users’ raw search activities (e.g. search logs) as well as those aggregated from many searchers (e.g. suggested queries). Therefore, we investigate whether we can reverse the process we have proposed in Section 3.3 and automatically construct a procedural knowledge base like wikiHow using search queries and relevant documents.
Table 3.6: Comparison of top 4 suggested search tasks produced by the proposed STS system and commercial search engines for example search tasks.

| Task: make blueberry banana bread |
|-------------------------------|---------------------------------|------------------------|
| PROPOSED                       | GOOGLE                          | BING                   |
| blend the mixture              | vegan blueberry banana bread    | best blueberry banana bread |
| mix flour                      | buttermilk blueberry banana bread | bisquick banana blueberry bread |
| add 1 egg                      | great blueberry banana bread recipe | banana blueberry cake |
| vanilla extract                | blueberry banana bread with yogurt | healthy banana blueberry bread |

<table>
<thead>
<tr>
<th>Task: slim down</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight loss</td>
</tr>
<tr>
<td>heavy food</td>
</tr>
<tr>
<td>junk food</td>
</tr>
<tr>
<td>keep up the mood</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task: play red alert2</th>
</tr>
</thead>
<tbody>
<tr>
<td>build a barracks</td>
</tr>
<tr>
<td>build a war factory</td>
</tr>
<tr>
<td>radar should</td>
</tr>
<tr>
<td>build a power plant/tesla reactor</td>
</tr>
</tbody>
</table>

related from search engines.

Problem 4 (Automatic procedural knowledge base construction from search activities, APKBC). Given a task $t$, we aim to (a) identify a search task $q$, and (b) collect $k$ related search tasks $p_1, \ldots, p_k$, and then (c) identify $n$ ($\leq k$) search tasks $p_{i_1}, \ldots, p_{i_n}$ to generate $n$ tasks $s_1, \ldots, s_n$ that can be performed to accomplish the task $t$ with text description.

Related search tasks can be identified from users’ search activities [18, 82, 128] and/or suggested by search engines [7, 25, 69, 99, 164]. Depending on the search intent, related actions, attributes, entities, or any form of reformulation of the original search task may be considered related, which however do not necessarily embed procedural knowledge. Therefore, the first challenge in extracting procedural knowledge is to identify the is-achieved-by relations, i.e., which of them correspond to the tasks that can achieve the goal. A textual description is a less ambiguous representation of a task in a procedural knowledge base, which can be accessed both by humans and by automatic decision support systems that require predefined natural language decision processes as input (e.g. QUADS [170]). Our second challenge is therefore to automatically generate text descriptions for procedural knowledge which attempt to conform to a common writing style. We summarize the interactions between task-oriented search and procedural knowledge base construction and illustrate its relation with search task suggestion problem (i.e. Problem 3) in Figure 44.
3.4.1 Modeling & Solution

Similar to Section 3.3.1, in this section, we propose a supervised learning framework, which requires zero extra manual annotation, and instead takes advantage of available artifacts; users’ search activities and manually curated procedural knowledge are used to train a model for the automatic procedural knowledge base construction problem. In particular, existing search logs can reveal how searchers have been trained to formulate queries, but they don’t necessarily or sufficiently cover how to search for procedural knowledge. On the other hand, existing procedural knowledge bases indicate how to accomplish tasks in a very comprehensive manner, but are not necessarily optimized for interactive search. We first describe how to extend the annotated corpus defined in Section 3.3.1 and further build a three-way parallel corpus for training, and then define linguistic and structural features for training sequence labeling models for both the search task suggestion problem and the automatic procedural knowledge base construction problem.

We create a three-way parallel corpus that contains triples \( \{q, t, c\} \) with a query \( q \), a matching task \( t \), and a context \( c \) of \( q \) and \( t \), which is a passage of text that describes the task. We build the three-way parallel corpus in an iterative manner.

We first identify the exact matching pairs of searchers’ issued search tasks and task descriptions in the procedural knowledge base, which guarantees precision in the first step, similar to the annotated corpus construction process, and can be achieved by scanning through the entire search query log to find each query \( q \) that matches the description of the task \( t \). We then retrieve the context \( c \) by extracting the textual content from the top documents relevant to the description of the task \( t \). We add the triple \( \langle q, t, c \rangle \) to the parallel corpus. We also collect related queries by combining the user-issued queries from the same session and the list of queries suggested by the search engine for the next iteration. If necessary, we may also use the summary of the task \( t \) to form an “artificial” search query \( q \) and issue it to the search engine to extract the context \( c \). Again, we exclude the triple \( \langle q, t, c \rangle \) that contains the original query \( q \) from the parallel corpus, which is only used for
Search queries in a session

- grow taller
- human growth hormone
- grow taller exercises

Task descriptions in procedural knowledge bases

Grow Taller

http://www.wikihow.com/Grow-Taller

- If you’re from a tall family and you’re not growing by your mid-teens, or if your height hasn’t changed much from before puberty to during puberty, then it’s a good idea to see a doctor....

- The human growth hormone (HGH) is produced naturally in our bodies, especially during deep or slow wave sleep. Getting good, sound sleep will encourage the production of HGH, which is created in the pituitary gland.

- ...There are tons of “grow taller” exercises on the Internet, which claim to help you grow....

Contexts retrieved from Web

- ... If you’re from a tall family and you’re not growing by your mid-teens, or if your height hasn’t changed much from before puberty to during puberty, then it’s a good idea to see a doctor....

- The growth hormone (HGH) is produced naturally in the pituitary gland during deep or slow wave sleep.

---

Figure 3.2: An example of the parallel corpus construction process

search engines to suggest real queries for the next iteration. Then, we compare the related queries for \( q \) with the known subtasks of \( t \). For each related query \( q' \), we first find the subtasks \( t_1, \ldots, t_n \) that contain \( q' \) in either its summary or explanation, and retrieve its context \( c' \). We then add all \( \{q', t_i, c'_i\}_{i=1}^{n} \) to the parallel corpus. We continue the iterative process.

Furthermore, we annotate the occurrence of the query \( q \) in the task \( t \)’s description. We also similarly annotate the occurrence of the summary and explanation of the task \( t \) in the context \( c \) by finding the contiguous sequence of words (a passage) from the context that is most relevant to the task summary or explanation. An example is shown in Figure 3.2. For training a sequence labeling model for automatic procedural knowledge base construction problem, we also employ the BIO encoding, i.e., each token in the task description (or context) that begins each occurrence of the query phrase (or the task summary or explanation) is labeled as \( B_Q \) (or \( B_{TS}, B_{TE} \)), the tokens that continue in the query phrase (or the task summary or explanation) are labeled as \( I_Q \) (or \( I_{TS}, I_{TE} \)), and all other tokens in the task (or the context) are labeled as \( O \).

We use the constructed parallel corpus, which contains triples of queries, task representations, and contexts, to train supervised models that can automatically create task-oriented queries for tasks, as well as task descriptions for task-oriented search tasks. We extract the same set of features (except the location feature) from both the context \( c \) (\( X_c \)) and task description \( t \) (\( X_t \)) for each triple in the parallel corpus, which together with the corresponding annotated BIO label (\( Y_c \) and \( Y_t \)) comprises a training instance for sequence labeling. We train a query construction model \( M_Q \) using the training sets (\( X_t, Y_t \)), a task summary construction model \( M_{TS} \) and a task explanation...
construction model \( M^{TE} \) using \((X^c, Y^c)\).

The training and prediction process for search task suggestion is the same as in Section 3.3.1. Similarly, to construct task descriptions for an unseen query \( q \), we first retrieve the context \( c \) and extract the features \( x^c \) from the context, and then apply the task description construction model \( M^c \) to extract each word sequence labeled as \( B^{TS}I^{TS} \ldots I^{TS} \) and \( B^{TE}I^{TE} \ldots I^{TE} \) as the summary and the explanation of a task \( t_i \) respectively from optimizing the following equation:

\[
y^c = \arg\max_{y^c \in \{B^{TS}, I^{TS}, O\}} p(y^c | x^c; M^{T\cdot}) \tag{3.3}
\]

Finally, given a task description \( t \), we directly apply the query construction model \( M^Q \) to extract a task-oriented search query \( q \) using Eq. 3.1 and then (similar to the process in building the parallel corpus) we identify the the queries related to \( q \) in both search logs and suggested queries. The entire search session (reformulation of queries, click activities, etc.) can be analyzed to determine how a specific user works to accomplish a task; therefore, by mining the search session data, we should be able to model how users typically accomplish tasks. As a result, the search engine should be able to correctly suggest to the user related tasks, rather than related entities or attributes. Although this assumption may not always hold in real-world search scenarios, it allows us to consider how related tasks can be further explored to extract subtasks for \( t \), as long as the task description construction model can identify any passage that can be used for the task summary or explanation.

For each related query \( p_i \), we collect its context by extracting relevant document snippets from search engines, and apply the task description construction model \( M^{T\cdot} \) to extract the most likely summary candidate \( w_i \) and explanation candidate \( v_i \) for task \( s_i \), according to Eq. 3.3, among all extracted summary candidates and explanation candidates for \( s_i \). We select the top-\( n \) tasks ordered by the estimated likelihood for their summaries.

### 3.4.2 Data Preparation

In addition to the two publicly available data sets (an English wikiHow data dump and the AOL search log) described in Section 3.3.2 we also leveraged public search engines (Google and Bing) to collect contexts. The data and code used in this section are available to download\footnote{https://github.com/ziy/pkb}.

We used Google to collect the contexts of the queries both in the parallel corpus (used for model training and testing) and those that are potentially useful for new procedural knowledge. We first extracted URLs for context candidates from their first search result pages, and excluded Web documents from the wikihow.com domain to increase the reliability and generalizability of the trained model, and also excluded google.com domains and URLs that have only domain names without subpaths, which are usually navigational search results. Then, we used the Boilerpipe\footnote{https://github.com/kohlschutter/boilerpipe Ver. 1.2.0} to extract textual content from the HTML documents. For queries in the parallel corpus, we finally downloaded 7,440 context documents and successfully processed 7,437 documents using Boilerpipe, and for end-to-end evaluation, we downloaded and extracted 3,512 context documents.
Table 3.7: Task summary generation results. A dagger (†) and one to three stars (⋆) represent significance levels of $p \leq 0.1, 0.05, 0.01, \text{ and } 0.001$ respectively, and ns represents not significant.

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Ma P</th>
<th>Ma R</th>
<th>Ma F1</th>
<th>Mi P</th>
<th>Mi R</th>
<th>Mi F1</th>
<th>R-2</th>
<th>R-S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF</td>
<td>.7576</td>
<td>.3260</td>
<td>.4207</td>
<td>.8524</td>
<td>.3344</td>
<td>.3455</td>
<td>.4463</td>
<td>.4392</td>
</tr>
<tr>
<td>HMM</td>
<td>.0042</td>
<td>.0008</td>
<td>.0014</td>
<td>.1213</td>
<td>.0010</td>
<td>.0015</td>
<td>.1017</td>
<td>.0660</td>
</tr>
<tr>
<td>SVM</td>
<td>.1464</td>
<td>.1116</td>
<td>.1175</td>
<td>.4585</td>
<td>.1041</td>
<td>.1119</td>
<td>.2425</td>
<td>.2301</td>
</tr>
<tr>
<td>LR</td>
<td>.1131</td>
<td>.0489</td>
<td>.0668</td>
<td>.4210</td>
<td>.0476</td>
<td>.0511</td>
<td>.1734</td>
<td>.1692</td>
</tr>
<tr>
<td>TFIDF</td>
<td>.0160</td>
<td>.1254</td>
<td>.0271</td>
<td>.0556</td>
<td>.1347</td>
<td>.0683</td>
<td>.1583</td>
<td>.1026</td>
</tr>
</tbody>
</table>

W/ subsets of features (+context)

| POS         | .5325 | .2881 | .3556 | .7093 | .2923 | .3153 | .3822 | .3788 |
| WORD        | .7738 | .2464 | .3279 | .9061 | .2433 | .2526 | .3435 | .3412 |

W/O subsets of features (+context)

| POS         | .7489 | .2651 | .3460 | .8813 | .2576 | .2695 | .3678 | .3645 |
| PAR         | .7723 | .3167 | .4129 | .8749 | .3088 | .3198 | .4170 | .4118 |
| WORD        | .5359 | .2945 | .3632 | .7084 | .3185 | .3408 | .3787 | .3704 |

Local       | .7590 | .2757 | .3755 | .8439 | .2734 | .2902 | .4053 | .4006 |
| Context     | .7736 | .3113 | .4081 | .8604 | .3159 | .3260 | .4069 | .4022 |

3.4.3 Experiment Settings

The baseline methods, feature extractors and learners are the same as those described in Section 3.3.3. When we evaluate our proposed end-to-end solution for the automatic procedural knowledge base construction problem, we compare with the current wikiHow knowledge base. In particular, we applied our procedural knowledge base construction approach to generate up to eight procedural descriptions and merged it with the wikiHow "steps". We manually judged whether each subtask summary and explanation that the system produced can be considered a valid "step" description for wikiHow. We also report the macro-averaged and micro-averaged Precision@8, and MAP averaged over all 50 test instances.

3.4.4 Experimental Results

We compare the results of our proposed approach with the baseline methods for task description (summary and explanation) construction in Tables 3.7 and 3.8. We first see that all the methods produce much lower F-1 and ROUGE scores for APKBC than those for STS, which suggests that task description generation is a difficult task, esp. explanation construction. Nevertheless, we can see that the proposed approach significantly outperforms all other classifiers in terms of F-1 and ROUGE metrics (at a significance level of $\leq 0.1$).

When comparing with the results from using each feature category, we see that not a single category can reach the same performance level as the proposed approach, which again implies that all
Table 3.8: Task explanation generation results. A dagger (†) and one to three stars (⋆) represent significance levels of $p \leq 0.1, 0.05, 0.01,$ and 0.001 respectively, and ns represents not significant.

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Ma P</th>
<th>Ma R</th>
<th>Ma F1</th>
<th>Mi P</th>
<th>Mi R</th>
<th>Mi F1</th>
<th>R-2</th>
<th>R-S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF</td>
<td>.4984</td>
<td>.3310</td>
<td>.3853</td>
<td>.7180</td>
<td>.3342</td>
<td>.3577</td>
<td>.3698</td>
<td>.3686</td>
</tr>
<tr>
<td>HMM</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000</td>
<td>.4724</td>
<td>.0050</td>
<td>.0050</td>
<td>.2450</td>
<td>.2324</td>
</tr>
<tr>
<td>SVM</td>
<td>.0053</td>
<td>.0026</td>
<td>.0034</td>
<td>.4911</td>
<td>.0012</td>
<td>.0018</td>
<td>.1737</td>
<td>.1659</td>
</tr>
<tr>
<td>LR</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000</td>
<td>.6978</td>
<td>.0000</td>
<td>.0000</td>
<td>.1470</td>
<td>.1408</td>
</tr>
<tr>
<td>TFIDF</td>
<td>.0002</td>
<td>.0015</td>
<td>.0003</td>
<td>.0062</td>
<td>.0103</td>
<td>.0103</td>
<td>.2657</td>
<td>.2498</td>
</tr>
<tr>
<td>POS</td>
<td>.1904</td>
<td>.0463</td>
<td>.0687</td>
<td>.7588</td>
<td>.0704</td>
<td>.0889</td>
<td>.1647</td>
<td>.1637</td>
</tr>
<tr>
<td>PAR</td>
<td>.7000</td>
<td>.0000</td>
<td>.0000</td>
<td>.9909</td>
<td>.0000</td>
<td>.0000</td>
<td>.1325</td>
<td>.1325</td>
</tr>
<tr>
<td>WORD</td>
<td>.4901</td>
<td>.3056</td>
<td>.3639</td>
<td>.7292</td>
<td>.2961</td>
<td>.3176</td>
<td>.3489</td>
<td>.3472</td>
</tr>
<tr>
<td>W/ subsets of features (+context)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS</td>
<td>.4722</td>
<td>.3198</td>
<td>.3695</td>
<td>.7206</td>
<td>.3411</td>
<td>.3554</td>
<td>.3580</td>
<td>.3554</td>
</tr>
<tr>
<td>PAR</td>
<td>.4837</td>
<td>.3169</td>
<td>.3718</td>
<td>.7274</td>
<td>.3208</td>
<td>.3468</td>
<td>.3804</td>
<td>.3793</td>
</tr>
<tr>
<td>WORD</td>
<td>.2546</td>
<td>.0247</td>
<td>.0410</td>
<td>.7575</td>
<td>.0194</td>
<td>.0292</td>
<td>.1614</td>
<td>.1614</td>
</tr>
<tr>
<td>Local</td>
<td>.1337</td>
<td>.0355</td>
<td>.0521</td>
<td>.6105</td>
<td>.0370</td>
<td>.0474</td>
<td>.1858</td>
<td>.1828</td>
</tr>
<tr>
<td>Context</td>
<td>.4534</td>
<td>.2879</td>
<td>.3359</td>
<td>.6823</td>
<td>.2723</td>
<td>.2922</td>
<td>.3532</td>
<td>.3500</td>
</tr>
</tbody>
</table>

the feature categories have contributed to the overall performance in task description construction problem. Comparing with using all but one feature categories, we can find that Word features are crucial for summary generation but not explanation generation, whereas POS and Parsing features are crucial for explanation generation but not summary generation. We also observe the non-Word features that contribute the most in Table 3.9. Although the performance does not match that of the query construction approach, we can still find interesting patterns in the results. First, we can see that nouns and verbs are still crucial in constructing task descriptions, and verbs (VB and VBP) rather than nouns are more likely selected as the beginning of a summary. To begin the task explanation, the word is expected to have a “Begin” indicator and/or a dependency type of nsubj, meaning the nominal subject of a sentence. Verb phrases are also important in identifying a task description.

Finally, we evaluated our proposed end-to-end APKBC solution, and compared it with wikiHow articles in Table 3.10. Due to the difficulty of the task description construction problem and lack of high-quality task-oriented search query candidates, we find that the automatic approach performs much worse than manual curation in building a brand new procedural knowledge base from scratch; nevertheless, we still find that the proposed approach is able to discover relevant subtasks that are not covered in the current wikiHow article, i.e. it can be used to improve the coverage and quality of existing procedural knowledge bases. Some examples are shown in Table 3.11. For example, for the task “sign up for airbnb”, one of the suggested queries “sign up for airbnb coupon” implies a coupon may be an important subsidiary resource of a task that the current wikiHow article does not know. For this particular task, a statement with the detailed coupon
Table 3.9: The non-Word features that contribute the most to each label in the automatic procedural knowledge base construction problem. “-1” presents the feature of the previous word, “VP”/“NP” represents if the word is inside a verb/noun phrase, “Begin” refers to the beginning of the text.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>O → B&lt;sup&gt;TS&lt;/sup&gt;</td>
<td><strong>POS</strong>: VB</td>
<td><strong>POS</strong>: VBP</td>
<td><strong>POS</strong>: NN</td>
<td><strong>DEP</strong>: appos</td>
<td><strong>POS</strong>: NNP</td>
</tr>
<tr>
<td>B&lt;sup&gt;TS&lt;/sup&gt; → I&lt;sup&gt;TS&lt;/sup&gt;</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: VB</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: VBP</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: NNP</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: NN</td>
<td><strong>DEP</strong>: case</td>
</tr>
<tr>
<td>I&lt;sup&gt;TS&lt;/sup&gt; → I&lt;sup&gt;TS&lt;/sup&gt;</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: VBP</td>
<td><strong>POS</strong>: NNP</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: IN</td>
<td><strong>DEP</strong>: xcomp</td>
<td><strong>POS</strong>: JJR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>O → B&lt;sup&gt;TE&lt;/sup&gt;</th>
<th>B&lt;sup&gt;TE&lt;/sup&gt; → I&lt;sup&gt;TE&lt;/sup&gt;</th>
<th>I&lt;sup&gt;TE&lt;/sup&gt; → I&lt;sup&gt;TE&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td><strong>POS</strong>: VBG</td>
<td><strong>POS</strong>: NN</td>
<td><strong>DEP</strong>: compound</td>
</tr>
<tr>
<td>VP</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: NN</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: DT</td>
<td><strong>NP</strong>:</td>
</tr>
<tr>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: NN</td>
<td><strong>VP</strong>:</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>: NNS</td>
<td><strong>POS&lt;sup&gt;-1&lt;/sup&gt;</strong>:</td>
</tr>
</tbody>
</table>

Table 3.10: Summary and explanation generation result for automatic procedural knowledge base construction

<table>
<thead>
<tr>
<th></th>
<th>Summary</th>
<th>Explanation</th>
<th>wikiHow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro P</td>
<td>.0997</td>
<td>.2046</td>
<td>.9677</td>
</tr>
<tr>
<td>Micro P</td>
<td>.0995</td>
<td>.2041</td>
<td>.9515</td>
</tr>
<tr>
<td>MAP</td>
<td>.0527</td>
<td>.1331</td>
<td>.9404</td>
</tr>
</tbody>
</table>

info is able to be extracted to describe the concrete task related to the resource. Even though this result may be from an advertisement Web page, it still delivers the freshest information that can hardly be added and updated instantly in manually created procedural knowledge bases.

We also summarize the errors into three major categories: (1) ambiguous task representations. For example, the “find a match” article on wikiHow<sup>18</sup> describes how to “look for a partner”, whereas search engines try to disambiguate the search intent by also suggesting queries in the context of data analysis such as “find a match in excel”, “find a match spreadsheet”, etc., (2) duplicated descriptions, and (3) quality of the text extracted from noisy Web documents. Moreover, as we note that not all wikiHow articles exactly follow the writing guide, the proposed approach can be used estimate the quality of each existing task description, similar to Eq. 3.2, and report potentially suspicious low-quality articles or spurious steps such as “You’re done.”, which violate the writing guidelines.

3.5 Procedural Knowledge Extraction from Unstructured Text

We plan to further study to extract procedural knowledge from unstructured text, such as Web documents and scientific publications, using richer syntactic, semantic, and discourse analysis, and studying the canonicalization of representation to facilitate analysis component construction and execution. The proposed algorithm is tested in case studies in Chapters 7 and 8.

<sup>18</sup>http://www.wikihow.com/Find-a-Match
Table 3.11: Partial suggested subtasks from the top 8 outputs produced by the proposed APKBC system for example tasks.

**Task: sign up for airbnb**
Airbnb is no longer running the $50 OFF $200 promo but you can still save $25 OFF Your First Airbnb Stay of $75 or more by copying and pasting this link into your browser: . . .

**Task: make blueberry banana bread**
Please don’t use regular whole wheat in this recipe – the loaf will turn out very dense
Add the wet ingredients - the egg mixture to the flour mixture and stir with a rubber spatula until just combined
If you’re in need of a quick, easy and delicious way to use up the ripe bananas in your house…definitely

**Task: become a cell phone dealer**
However, the cell phone provider may place restrictions on the manner in which you can use its company name, phone brands and images
Visit the state’s business licensing agency’s website and your city’s occupational/business licensing department’s website to determine if you need a license for your prepaid cell phone business

3.6 Procedural Knowledge Extraction from Community QA Site

We plan to further study to extract procedural knowledge from online community question answering sites, taking the advantage of their unique features, such as users’ feedback, comments, etc. The proposed algorithm is tested in Chapter 8 as well as open shared task related to cQA.

3.7 Proposed Work

We plan to extend our prior work on leveraging procedural knowledge for task-oriented search by further exploring a wider range of human-generated data sources, such as on-line community question answering sites, bibliographic databases, etc., and studying the canonicalization of representation to facilitate analysis component construction and execution.
Chapter 4

Exploration Algorithms

This chapter studies the optimization problem in analytics meta learning – analytics space exploration under limited resource constraint, which aims to rapidly identify the optimal analytics procedure.

In Section 4.1, we review prior work related to the modeling and optimization problem. Then in Section 4.2, we first study configuration space exploration, a subproblem of analytics space exploration that considers only the information pipelines that have fixed analytics procedures, i.e. those that contain only execution processes, and present an exploration strategy based on hierarchical Bayesian model and stochastic scheduling. We report the experimental results on our preliminary study using the TREC Genomics QA task in Section 4.3, which involves a configuration space (a subspace of an analytics space) consisting of trillions of possible execution processes. In Section 4.4, we propose the general exploration strategy for arbitrary analytics spaces. In Section 4.5, based on the literature review on overflow scheduling and planning, we propose additional exploration strategies, and conduct a theoretical comparison across these strategies, such as the rationality of their assumptions, the asymptotic property and the optimality of the optimization algorithms, etc.

4.1 Related Work

4.1.1 Automatic Service Composition

Rao [129] has surveyed the workflow composition algorithms or AI planning algorithms to tackle the remaining automatic service composition problem. We further analyze the “intelligent layer” behind these algorithms, which may include Bayesian modeling, graph algorithms, etc.

4.1.2 Limitations of Automatic Service Composition Solutions

These automatic composition methods do not optimize the system performance for the analytics tasks, or the optimization could easily become intractable as the system gets complicated.
Excessive requirement of manual configuration. Since most automatic composition approaches only deal with configured components, users have to predetermine the values to the configuration parameters. Similarly, most composition approaches include an evaluation process to select the top composition strategies among those fulfilling the requirement, which also requires specification of additional non-functional attributes of each component such as users’ preference \([129]\). After all, the performance depends heavily on what values the user assigns to these parameters, which is, however, clueless, time-consuming, and subjective, and thus unreliable. When it comes to constructing an intelligent information system, we can, fortunately, capture a component’s utilities by its “intelligence” (benefit) and its resource consumption (cost), both of which can be measured against a benchmark during task execution. A fully automated composition approach should configure these attributes itself.

Curse of dimensionality. The number of possible configurations of a typical intelligent information system can increase exponentially \([169]\), which implies that any exhaustive search based method may fail to produce an optimal solution if finite resource can be allocated to perform composition. The solution should efficiently discover the best configuration without trying all combinations.

Lack of domain adaption support. Moreover, existing automatic composition approaches make the assumption that each system has no relation with any past existing system, and thus build each information systems entirely from scratch. In reality, development of a solution to a new analytics task could be inspired by, and further based on or extended from, an existing solution to a similar task, which also suggests that the composition efficiency can be improved largely by domain adaptation.

4.1.3 Automatic Optimization of Configurable Systems

Optimization of configurable systems are also studied in the context of automation and assistant of software construction. An example work is by \([141]\). We further investigate the applicability of such proposals and their algorithms.

[PROPOSED WORK, C.F. SECTION 4.5 AND CHAPTER 9]

4.1.4 Optimization Theories

The proposed problem is similar to multi-armed bandit assumption, which has been extensively studied and applied in many applications.

[PROPOSED WORK, C.F. SECTION 4.5 AND CHAPTER 9]

4.1.5 Automatic Evaluation of Software

Although evaluation is involved in automatic service composition, it is more widely discussed in the context of general software development. Traditionally, version evaluation in an iterative development process \([80]\) is done by manual judgment, with each iteration, based on how well each integrated component satisfies the task goal, and planning is done by manual prioritization of
changes that are most needed in the next iteration based on their priorities and schedule estimates. This process has been partially automated or assisted by test automation and Continuous Integration (CI) tools [102, 146], which have been widely adopted in most modern development processes such as extreme programming [12], where a set of predefined test cases are executed automatically after each code commit and the actual outcomes are compared with the output specification to check if the requirements are met after the change. However, most test automation frameworks can only serve as a sanity checker for intelligent information systems, since their performance on a given analytics task is usually judged subjectively on a graded or continuous scale, rather than binary “pass or fail”. Machine learning community has also attempted to make the runtime performance of individual analysis components easily compared between various implementations and versions and accessed by the developers, e.g. MLComp [1], ml-benchmarks [2], OpenML [3], and further integrate the automatic benchmarking system into an automated build process, such as Jenkins CI framework [4].

4.2 Configuration Space Exploration

In this section, we consider a simpler problem – configuration space exploration, which aims to optimize the configuration space for information pipelines with a fixed analytics procedure. We propose a novel formulation and solution to the associated constraint optimization problem of analytics meta learning (cf. Problem 2), which comprises two sub-problems: component characteristics estimation and stochastic scheduling.

As we introduced in Section 2.3, the utility of an information system can be measured by runtime performance and task performance, and in the constraint version of the analytics meta learning problem (cf. Problem 2), the total cost of executing the entire analytics space is also measured. In order to estimate the performance at the configured component level, we introduce notations \( c(⟨f, ω⟩, x) \), \( b(⟨f, ω⟩, x) \), and \( u(⟨f, ω⟩, x) \) to capture two important characteristics of the configured component \( ⟨f, ω⟩ \): the cost of resource required to execute the component on input \( x \), the benefit of executing the configured component to performance improvement, and the utility of the configured component, which can be defined as a function of cost and benefit, or

\[
u(⟨f, ω⟩, x) = u(b(⟨f, ω⟩, x), c(⟨f, ω⟩, x)).
\]

The benefit of a single component is relatively difficult to measure without being integrated into a system with other components, where we can leverage commonly-used evaluation metrics (cf. Definition 11) for information systems, e.g. F-1 and MAP, etc. In information pipelines, we simply assume a configured component shares the benefit with the component that follows, if no direct measurement is available; the true cost or benefit of a configured component is estimated based on the cumulative effect over trillions of pipeline combinations.

We extend the concepts of cost and benefit for a configured component \( \langle f, \omega \rangle \) to a system \( s \) and an analytics space \( A \): the cost to execute a system is the sum of costs to execute each configured component, and the performance of a system corresponds to the final output from last execution. They can be formally defined as follows:

\[
\begin{align*}
c(s, x) &= \sum_{t=1}^{n} c(\langle f, \omega \rangle_t, x_t) \quad (4.1) \\
b(c, x) &= b(\langle f, \omega \rangle_n, x_n) \quad (4.2)
\end{align*}
\]

where we assume the system \( s \) consists of \( n \) consecutive steps, \( \langle f, \omega \rangle_t \) is the \( t \)-th configured component in the corresponding execution process, and \( x_t \) represents the input of \( \langle f, \omega \rangle_t \), which accumulates information gathered from all previous steps. The utility of the configured component can also be defined as a function of cost and benefit, or

\[
u(s, x) = u(b(s, x), c(s, x)).
\]

The cost of the entire analytics subspace is defined as the sum of unique executions of configured components on all outputs from previous steps. The benefit or utility of the analytics space is defined as that of the best-performing system.

\[
\begin{align*}
c(A, x) &= \sum_{t=1}^{n} \sum_{c_1=1}^{c_1} \cdots \sum_{c_t=1}^{c_t} c(\langle f, \omega \rangle_t^{c_t}, x_t^{c_t}(c_1, \ldots, c_{t-1})) \quad (4.3) \\
b(A, x) &= \max_{s \in A} b(s, x) \quad (4.4) \\
u(A, x) &= \max_{s \in A} u(s, x) \quad (4.5)
\end{align*}
\]

where \( \langle f, \omega \rangle_t^{c_t} \) represents the \( c_t \)-th option at step \( t \), and we use \( x_t^{c_t}(c_1, \ldots, c_{t-1}) \) to distinguish the input to the configured component \( \langle f, \omega \rangle_t^{c_t} \) produced from upstream trace \( \langle f, \omega \rangle_1^{c_1} \rightarrow \ldots \langle f, \omega \rangle_t^{c_{t-1}} \). We see that \( c(A, x) \neq \sum_{s \in A} c(s, x) \). In fact, if an information pipeline \( s \) has been executed, and another information pipeline \( s' \) has the same prefix, i.e.,

\[
\langle f, \omega \rangle_1^{c_1} \rightarrow \ldots \langle f, \omega \rangle_{t-1}^{c_{t-1}} = \langle f, \omega \rangle_1^{c_1} \rightarrow \ldots \langle f, \omega \rangle_{t-1}^{c_{t-1}}
\]

for some \( t \), then we do not need to repeat the executions for the configured components along the prefix, which is one of the key ideas for the problem solution and implementation.

The analytics meta learning problem description (cf. Problem 2) appears isomorphic to constrained optimization. However, in the analytics meta learning case, both cost and benefit of a configured component are unknown to the model until the component is executed. In the next three sections, we describe an exploration strategy to the configuration space exploration problem.

### 4.2.1 Solution Overview

In our initial case study, we adopt the constraint that each parameter must have a finite number of values, in order to support an efficient practical implementation and experimentation.\(^5\) Moreover,

\(^5\)To incorporate continuous parameters, one can discretize the feasible region by equally binning the region or sampling values from it. Directly tackling an optimization problem with continuous parameters requires a better
we note the problem in this section focuses on determining the globally optimal system general
to all inputs, whereas researchers have observed that some components or configurations might
improve the system performance on only some types of inputs while hurt the performance on other
types \[8\]. Accordingly, one can further extend the solution to a variant also incorporating the
dependency on the input type\[6\].

We note the constraint optimization problem (cf. Problem 2) is trivial if the total cost to it-
\[eratively execute all the components and configurations for all inputs does not exceed available
processing capacity. However, this is not the case for most real world problems, since the num-
ber of systems and executions grow exponentially as the number of steps increases. A typical
information processing pipeline consisting of 12 components is shown in Section 4.3; with up to
four parameters per component, and up to six options each parameter, there would be an estimated
\[6.050 \times 10^{13}\] executions if all the unique systems were evaluated.

In an ideal case where (1) the exact cost \(c(\langle f, \omega \rangle, x)\) of each configured component on each
input is known, (2) the benefit \(b(\langle f, \omega \rangle, x)\) of each configured component \(\langle f, \omega \rangle\) is an \(i.i.d\). random
variable, and (3) the utility only considers the benefit, i.e. \(u(\langle f, \omega \rangle, x) = b(\langle f, \omega \rangle, x)\), the optimal
solution to this problem can be yielded by adding configured components and inputs in descending
order of least cumulative cost (LCC), which is defined as the sum of the component’s original
cost plus minimal additional cost to execute the components down the pipeline, and formulated as
follows:

\[
\text{LCC}(\langle f, \omega \rangle_t, x) = c(\langle f, \omega \rangle_t, x) + \sum_{u=t+1}^{n} \min_{d=1}^{m_u} c(\langle f, \omega \rangle_{u,d}, x) \tag{4.6}
\]

When we relax the first assumption used for the simplified case (the cost is revealed until the
execution is done), the approaches for deterministic scheduling problems \[72\] do not provide an
adequate solution. Instead, we refer to the stochastic scheduling problem \[36\] or stochastic knap-
sack problem \[33\], which makes a different assumption that job durations are random variables
with known probability distributions. These approaches provide a more general formulation suit-
able for the configuration space exploration problem.

We also relax the second assumption and assume the performance of a component is not \(i.i.d\).
For example, a weak configured component \(\langle f, \omega \rangle\) (e.g., naive approach, or buggy code) tends to
exhibit low performance for every input \(x\); on the other hand, a complex input \(x\) (e.g., a very diffi-
cult question posed to a question answering system) may have a negative effect on the performance
of all configured components. We are inspired to leverage dependency of characteristics and prior
knowledge to dynamically (and more wisely) select which system to evaluate next from the pool.
We can utilize the execution history to estimate \(c(\langle f, \omega \rangle, x)\) and \(b(\langle f, \omega \rangle, x)\) by establishing a
dependency between them. Intuitively, components that previously showed high benefit (or low
cost) tend to achieve again high benefit (or low cost) combined with components from other steps,
and components of low benefit (or high cost) tend to be pruned from the pool. In addition, we
understanding of how the behavior of the optimization objective varies as the parameter changes (e.g., linearity, con-
vexity, etc.), and further relies on various continuous optimization techniques. Extending the exploration strategy for
continuous parameter values is left to future work.

\[6\]As the intermediate data are persisted, a post-performance analysis can be easily conducted to identify the corre-
lation between the input type and the system performance.
can also estimate the cost from prior knowledge, e.g., retrieving relevant passages usually requires more resources than tokenizing question texts in a retrieval system. We may also relax the third assumption based on our estimation of cost and benefit, which is left for future work.

We develop the solution to this problem based on hierarchical Bayesian modeling, where both prior knowledge and previous execution history are incorporated into the same framework. The basic idea is that we associate each cost $c(⟨f, ω⟩, x)$ or benefit $b(⟨f, ω⟩, x)$ with an unknown distribution, with parameters representing our knowledge and past observations. Starting with some known priors, we select the system to execute based on the greedy algorithm for the stochastic knapsack problem, and then update the priors each time we finish an execution. The process repeats until $C$ is reached. We first discuss how to estimate the cost and benefit of each component to solve the stochastic scheduling problem in Sections 4.2.2 and 4.2.3, and show an intuitive example in Section 4.2.4.

### 4.2.2 Modeling cost and benefit distributions

We apply a hierarchical Bayesian model to capture the component characteristics hierarchically inherited from global level and step level. Specifically, we let two hyperparameters $γ$ and $β$ denote the average cost and benefit of a configured component globally across all steps and inputs, and then for each configured component $⟨f, ω⟩$ in the $t$-th step, we introduce step-level parameters $γ_t$ and $β_t$ to capture the characteristics shared by the components in the $t$-th step, and component-level parameters $γ_c^t$ and $β_c^t$ to model the performance of $⟨f, ω⟩$ regardless of the specific input $x$.

Analogous to $γ_t$ and $β_t$, we introduce parameters $γ^x$ and $β^x$ to denote the average cost and benefit of all configured components for each input $x$, which indicates the difficulty of processing each input. Given all the hyperparameters, the random variable $c(⟨f, ω⟩, x)$ or $b(⟨f, ω⟩, x)$ is the outcome from a series of generative processes from $γ$ or $β$, which hence defines a joint probability distribution, shown in Figure 4.1.

Based on prior knowledge of expected cost and benefit, we can assign values for global hyperparameters $γ$ and $β$, and optionally for some step-level hyperparameters $γ_t$ and $β_t$. Subsequently, unspecified step-level hyperparameters and configured component level parameters $γ_c^t$ and $β_c^t$ can

---

7 Due to the discrete configuration assumption, the same component with a different configuration is treated as a different component. One can easily extend the model to incorporate the dependency between the configured components and their base components.
be estimated with *maximum a posteriori* method based on priors $\gamma$ and $\beta$, and all observations (corresponding to previous execution records). For configured components that have not been executed, $c(\langle f, \omega \rangle_t^c, x)$ and $b(\langle f, \omega \rangle_t^c, x)$ can be predicted from estimated parameters $\gamma_t^c$, $\gamma^c$ and $\beta_t^c$, $\beta^c$ with Bayesian inference method.

### 4.2.3 Optimization via stochastic scheduling

Once $c(\langle f, \omega \rangle_t^c, x)$ and $b(\langle f, \omega \rangle_t^c, x)$ are estimated for the configured components in the pool, we can rewrite the objective for the deterministic scheduling problem (Equation 2.2) to a stochastic variant (a.k.a. stochastic knapsack problem [33]), which assumes that job durations and/or rewards are random variables with known probability distributions, as follows:

$$\max_{A^t \subseteq A \cap (X,t) \subseteq (X,Y)} E_u(A' \mid X', Y') \quad \text{s.t.} \quad E_c(A', X') < C \tag{4.7}$$

where $E_u$ and $E_c$ represent the expected utility and cost over the random variables $u(\langle f, \omega \rangle_t^c, x)$ and $c(\langle f, \omega \rangle_t^c, x)$ respectively.

Among the solutions (or scheduling policies) for the stochastic scheduling problem, *adaptive policies* have been studied extensively in the literature, which dynamically choose which systems to execute next based on previously executed systems. In general, they can achieve a better approximation ratio to the optimal solution [33], and can also be naturally integrated with the configuration space exploration problem, since the estimation of all parameters and distributions of $b(\langle f, \omega \rangle_t^c, x)$ and $c(\langle f, \omega \rangle_t^c, x)$ are dynamically updated as execution proceeds. A commonly adopted greedy policy for the (stochastic) knapsack problem is to sort the items by

- decreasing (expected) benefit,
- increasing (expected) cost,
- decreasing (expected) benefit density, or
- $b(\langle f, \omega \rangle_t^c, x)/\text{LCC}(\langle f, \omega \rangle_t^c, x)$

in the context of analytics meta learning problem. Finally, based on the benefit evaluated for each system $\langle f, \omega \rangle_t^c$ and input $x$, we select the systems by further measuring their generalizability to new data not yet realized, which can be estimated by various model selection methods, e.g. cross-validation, bootstrap, etc.

We present the solution in Algorithm 1 where $\langle P \rangle$ represents the system subspace spanned by the component pool $P$, $h(\langle f, \omega \rangle_t^c, x)$ is a heuristic function defined on $\langle f, \omega \rangle_t^c$ and $x$, to prioritize the components. Examples of $h$ function can be

- cost relevant ($-E_c[\text{LCC}(\langle f, \omega \rangle_t^c, x)]$),
- benefit relevant ($E_0[b(\langle f, \omega \rangle_t^c, x)]$),
- benefit density ($E_0[b(\langle f, \omega \rangle_t^c, x)]/E_c[\text{LCC}(\langle f, \omega \rangle_t^c, x)]$) or
- profit ($E_0[b(\langle f, \omega \rangle_t^c, x)] - \lambda E_c[\text{LCC}(\langle f, \omega \rangle_t^c, x)]$, etc.

Algorithm 1 applies an adaptive policy to execute the trace (lines 6–7), predict the random variables (line 3), and reestimate parameters (line 8) inside the loop (lines 2–8). This framework assumes no prior beliefs, nor probability distributions of component characteristics, which can be customized
Algorithm 1: Greedy algorithm

**input:** Hyperparameters $\gamma$, $\beta$ and capacity $C$

**output:** Optimal system $s^*$

1. $X \leftarrow \emptyset$, $P \leftarrow \emptyset$, $Q \leftarrow \{(f, \omega)_{t,x}^c\}_{t,c,x}$
2. While $c(A, X) < C$ do
   3. For each $(f, \omega)_{t,x}^c \in Q$ do predict $b((f, \omega)_{t,x}^c)$, $c((f, \omega)_{t,x}^c)$, LCC$((f, \omega)_{t,x}^c)$;
   4. $(f, \omega)_{t,x}^c = \arg\max (f, \omega)_{t,x}^c \in Q h((f, \omega)_{t,x}^c)$;
   5. $Q \leftarrow Q \setminus \{(f, \omega)_{t,x}^c, x'\}$, $X \leftarrow X \cup \{x'\}$, $P[t] \leftarrow P[t] \cup \{(f, \omega)_{t,x}^c, x'\}$;
   6. For each $x \in X$ and $s \in \langle P \rangle$ do
      7. If $s$ has not been executed for $x$ then execute to instantiate $c(s, x)$ and $b(s, x)$;
   8. For each $t$ and $\langle f, \omega \rangle_t^c \in A$ do update $\gamma_t^c, \gamma, \beta_t^c, \beta$;

9. $s^* = \arg\min_{s^* \in A} E_{\mathcal{X}}[b(s, x)]$;

according to specific tasks. One can also consider to change the strategy function $h$ over time, e.g. promoting configured components with greater benefit variance or greater cost once the exploration is stuck in a local minimum. Observing how different strategy functions affect the exploration process is left for future work.

### 4.2.4 An example: 3-stage linear characteristics

To motivate our solution, we assume that the component characteristics follow a simple and intuitive 3-stage linear model [87], where each observation and hyperparameter follows a Gaussian distribution, with mean drawn from a distribution parameterized by the upper level in the hierarchy and variance predefined and fixed. In the simplified 3-stage linear model, we assume each input $x$ to each configured component $(f, \omega)_t^c$ follows uniform distribution over all possible inputs. The generative processes for $c((f, \omega)_t^c, x)$ and $b((f, \omega)_t^c, x)$ can therefore be instantiated from the general solution framework (Figure 4.1) and formulated as follows:

$$c((f, \omega)_t^c, x) \overset{\text{iid}}{\sim} \mathcal{N}(\gamma_t^c, \tau_t^2)$$

$$\gamma_t^c \overset{\text{iid}}{\sim} \mathcal{N}(\gamma, \tau^2)$$

$$b((f, \omega)_t^c, x) \overset{\text{iid}}{\sim} \mathcal{N}(\beta_t^c, \sigma_t^2)$$

$$\beta_t^c \overset{\text{iid}}{\sim} \mathcal{N}(\beta, \sigma^2)$$

We derive the posterior distributions for $\gamma_t^c$, $\gamma_t$ and $\beta_t^c$, $\beta_t$, which can be used to update the hyperparameters (line 8 in Algorithm 1). All the posterior probabilities follow Gaussian distribution...
Table 4.1: Posterior distributions for each of $\gamma^c_t$, $\gamma_t$ and $\beta^c_t$, $\beta_t$ in modeling analytics space exploration.

<table>
<thead>
<tr>
<th></th>
<th>$\theta$</th>
<th>$\phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma^c_t$</td>
<td>$\tau^c_t^{-2}n^c_t \bar{c}^c_t + (\tau^c_t^2 + \tau^2)^{-1}\gamma$</td>
<td>$\tau^c_t^{-2}n^c_t + (\tau^c_t^2 + \tau^2)^{-1}$</td>
</tr>
<tr>
<td>$\gamma_t$</td>
<td>$\sum_c(\tau^c_t^{-2}n^c_t \bar{c}^c_t + (\tau^c_t^2 + \tau^2)^{-1}\gamma$</td>
<td>$\sum_c(\tau^c_t^{-2}n^c_t + (\tau^c_t^2 + \tau^2)^{-1}$</td>
</tr>
<tr>
<td>$\beta^c_t$</td>
<td>$\sigma^c_t^{-2}n^c_t \bar{b}^c_t + (\sigma^c_t^2 + \sigma^2)^{-1}\beta$</td>
<td>$\sigma^c_t^{-2}n^c_t + (\sigma^c_t^2 + \sigma^2)^{-1}$</td>
</tr>
<tr>
<td>$\beta_t$</td>
<td>$\sum_c(\sigma^c_t^{-2}n^c_t \bar{b}^c_t + (\sigma^c_t^2 + \sigma^2)^{-1}\beta$</td>
<td>$\sum_c(\sigma^c_t^{-2}n^c_t + (\sigma^c_t^2 + \sigma^2)^{-1}$</td>
</tr>
</tbody>
</table>

of the form $\mathcal{N}(\phi^{-1}\theta, \phi^{-1})$, and the values of $\theta$ and $\phi$ are listed in Table 4.1 for each parameter, where $n^c_t$ represents the number of times $f^c_t|\omega^c_t$ has been executed thus far, and $\bar{c}^c_t$ and $\bar{b}^c_t$ are the average cost and benefit of the component in the history. We see that at the beginning of the experiment, when $n^c_t = 0$ for all $t$ and $c$, each parameter takes the same value from the user specified hyperparameter ($\gamma$ or $\beta$), and as the experiment proceeds, each parameter is dynamically shifted away from the value of hyperparameter to better model the execution behavior of the step or component ($\bar{c}^c_t$ and $\bar{b}^c_t$).

Finally, we define function $h$ (line 5 in Algorithm 1) as the profit, i.e.

$$E_b[b(\langle f, \omega \rangle^c_t, x)] - \lambda E_c[LCC(\langle f, \omega \rangle^c_t, x)]$$

for the preliminary experiments. Equivalently, we can dynamically prune the components of either high cost or low benefit. In Section 4.3, we illustrate how the model was implemented in the framework to tackle a real-world analytics task.

### 4.3 Preliminary Case Study: TREC Genomics Passage Retrieval

In this section, we implement the proposed exploration strategy and apply the configuration space exploration framework to the problem of building an effective biomedical document and passage extraction system from available components and component options. The goal of this work is to demonstrate the effectiveness of the proposed approach and the current open source implementation. Specifically, we employed the topic set and benchmarks from the question answering task of TREC Genomics Track [59], as well as commonly-used tools, resources, and algorithms cited by participants. A set of basic components was selected and adapted by writing wrapper code where necessary. Then an experiment configuration descriptor was defined for the resulting set of configured components. This configuration space was explored with the proposed strategy automatically, yielding an optimal and generalizable configuration which outperformed published results of the given components for the same task.

#### 4.3.1 Task & data set description

The TREC Genomics QA task involves retrieval of short passages that specifically address an information need expressed as a question, and the answers are provided along with a reference to the
Table 4.2: Summary of analysis components in TREC Genomics QA system

<table>
<thead>
<tr>
<th>Category</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP tools</td>
<td>LingPipe HMM based tokenizer</td>
</tr>
<tr>
<td></td>
<td>LingPipe HMM based POS tagger [100]</td>
</tr>
<tr>
<td></td>
<td>LingPipe HMM based named entity recognizer</td>
</tr>
<tr>
<td></td>
<td>Rule based lexical variant generator [1, 140, 148, 162]</td>
</tr>
<tr>
<td>KBs</td>
<td>UMLS for syn/acronym expansion [35, 140, 155, 173]</td>
</tr>
<tr>
<td></td>
<td>EntrezGene for syn/acronym expansion [35, 93, 148, 162, 173]</td>
</tr>
<tr>
<td></td>
<td>MeSH for syn/acronym expansion [37, 148, 162, 173]</td>
</tr>
<tr>
<td>Retrieval tools</td>
<td>Indri system [100, 140, 143, 155]</td>
</tr>
<tr>
<td>Reranking</td>
<td>Important sentence identification [14, 37, 153, 155, 162, 173]</td>
</tr>
<tr>
<td>algorithms</td>
<td>Term proximity based ranking [14, 155]</td>
</tr>
<tr>
<td></td>
<td>Score combination of different retrieval units [140, 148, 155]</td>
</tr>
<tr>
<td></td>
<td>Overlapping passage resolution</td>
</tr>
</tbody>
</table>

location of the answer in the original source document. A total of 64 genomics-related topics were developed for this task, asking about biological processes or relationships between biomedical entities. We employed both the document collection and the topic set from the official evaluation and focused on two task-specific metrics: DocMAP and PsgMAP. Intuitively, DocMAP measures the relevance of the documents retrieved by the system, regardless of the relevance and conciseness of the passages extracted. The PsgMAP metric also considers the relevance of extracted passage spans. Details can be found in the overview paper [59].

To evaluate the generality of the framework in selecting optimal systems for novel data, and further test the performance of the selected systems, we implemented nested K-fold cross-validation (leave-one-out, 5-fold and 10-fold) and nested bootstrap methods, where two subsets of 28 topics used by TREC Genomics 2006 were held out for system selection (validation set) and performance evaluation (test set) respectively. Furthermore, we employed 36 topics from TREC Genomics 2007 to test the adaptability of systems for a similar but slightly different task, with questions asking for lists of specific biomedical entities (in Section 4.3.6).

4.3.2 Analysis component construction: resource and algorithm integration

We mainly considered four different aspects when collecting resources and implementing algorithms to build a biomedical question answering system: **NLP tools**, **Knowledge Bases (or KBs)**, **retrieval tools**, and **reranking algorithms**. We considered popular and successful approaches reported in the TREC Genomics literature to explore. We summarize these components in Table 4.2 and briefly describe them in the rest of this subsection.

First, many successful systems employed natural language processing of the questions and/or target texts, using algorithms, toolkits, and pre-trained models for sentence segmentation, tokenization, part-of-speech tagging, named entity recognition, etc. To establish a benchmark for NLP in our system, we focused on several rule-based approaches for generating lexical variants,
and supervised learning methods provided by LingPipe.

Second, tens of biomedical KBs have been organized and maintained. Some involve knowledge from all areas of biomedical research, while others focus on only a few subareas, e.g., disease, genomics, etc. The implication is that for a particular topic, the quality of the answer generated by the system may vary by the KB(s) that are used. We focused on three resources most popular with TREC Genomics participants: UMLS, EntrezGene, and MeSH.

Next, to retrieve relevant documents from the unstructured biomedical corpus, we need to rely on a widely-used open-source search engine, e.g. Indri. Finding an optimal retrieval configuration requires selecting parameter values specific to each search engine, such as the retrieval model and the parameters in the scoring function, smoothing method, query formulation strategy, etc.

Finally, a number of participants performed postprocessing on their passage output. This was done to refine the boundaries and ranks of retrieved passages, using techniques such as evidence targeting, answer merging and rescoring, etc. The aspect that categorizes these approaches is reranking algorithms.

We followed the basic pipeline steps for a question answering system [151] and implemented a domain-independent QA framework. We integrated benchmarks, task-specific evaluation methods, as well as 12 components, and specified one to several values for each parameter associated with each component, and plugged them into the framework. To support straightforward reproducibility of results, the frameworks, components, and support materials are available online as part of our open source release.

4.3.3 Experimental results

We designed two experiments, given moderate and large-scale configurations respectively. In the first experiment, we limited the number of options and consequently derived 32 configured components ($A$), which could produce a maximum number of 2,700 systems and require 190,680 individual execution steps to process all the questions. An experiment of this complexity was expected to execute within a day on our available test hardware (corresponding to $C$ with resource defined by execution time). The mean and standard deviation for global execution time were initially set as 10 seconds ($\gamma$ and $\tau$) and the expected benefit for a single configured component was set as 0.1 in terms of PsgMAP ($\beta$ and $\sigma$), and the parameters at step and component levels were estimated by an empirical Bayesian method. In the second experiment, we planned to test the scalability of the implementation by aggressively setting up to six values for each parameter, and thus yielded an analytics meta learning problem with 2,946 configurations and $1.426 \times 10^{12}$ systems, requiring $6.050 \times 10^{13}$ executions in total to evaluate the entire space.

We compare the settings for the two experiments with the official TREC 2006 Genomics test results for the participating systems in Table 4.3. Although many more configured components were implemented by the original participants, only 92 different systems were evaluated. We estimate the number of components, configurations and executions evaluated by the official test, and show the evaluation results [60] in Table 4.3 for reference. The results reported from the

*http://www.lemurproject.org/indri/*
two experiment settings were evaluated with the nested leave-one-out cross-validation method. Different strategies of estimating generalizability of systems are further compared in Section 4.3.6.

From Table 4.3, we can see that the best system derived automatically by the proposed approach outperformed the best participating system in terms of both DocMAP and PsgMAP, with fewer, more basic components. Similarly constrained exploration of the much larger Scaled AML configuration failed to yield a system that performs as well in comparable time, highlighting the importance of a priori human selection of appropriate configurations and/or hyperparameter values. Nevertheless, the configuration yielded by exploring the larger space at the same capacity cost still outperformed most of the participating systems reported in the TREC Genomics track paper. The detailed pipeline configuration of the best system derived by the exploration strategy is described in Section 4.3.4 as the baseline to analyze component contributions.

### 4.3.4 Component analysis

As the best system has been discovered, we report the performance of systems with only a single component or configuration different from the best system, in terms of PsgMAP, to provide a useful comparison. In this section, we investigate in detail how each component or configuration contributes to overall system performance. The results are shown in Table 4.4 for nominal configurations, e.g. different KBs, score transformation methods, and Figure 4.2 for real-valued configurations, e.g. smoothing parameters, term weights, etc. We also report the significance test (t-test) results, and label the scores with different significance levels, in Table 4.4. The configuration of the best system is also shown in Table 4.4, which uses most available resources (KBs, reranking algorithms, NLP tools).

We see that the performance of leveraging various sources varied for synonym expansion and acronym expansion. Incorporation of MeSH in synonym expansion could make the biggest contribution to the performance, while both UMLS and EntrezGene also benefited the system’s perfor-
Table 4.4: Variation of nominal configurations for all components, including synonym and acronym expansion, lexical variants, sentence extraction, proximity based ranking, filtering of overlapping or identical passages, and score transformation. Significance levels of 0.1, 0.05, 0.01, and 0.005 are indicated by a sharp symbol (♯) and one to three star symbols (⋆) respectively. Best baseline system: resolve acronym by UMLS; filter identical passages; use lexical variants; use proximity based ranking with synonyms; apply sentence extraction with synonym; resolve synonym by MeSH+EntrezGene; combiner scores transformed by reciprocal rank.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>DocMAP</th>
<th>PsgMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best baseline system</td>
<td></td>
<td>.5257</td>
<td>.1744</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>None</td>
<td>.4169**</td>
<td>.1256*</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>UMLS</td>
<td>.4619‡</td>
<td>.1439</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>MeSH</td>
<td>.5050</td>
<td>.1710</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>EntrezGene</td>
<td>.4302*</td>
<td>.1233*</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>UMLS+MeSH</td>
<td>.4942</td>
<td>.1609</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>UMLS+EntrezGene</td>
<td>.4639‡</td>
<td>.1405</td>
</tr>
<tr>
<td>Synonym expansion with KBs</td>
<td>UMLS+MeSH+EntrezGene</td>
<td>.4959</td>
<td>.1603</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>None</td>
<td>.5253</td>
<td>.1751</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>MeSH</td>
<td>.5226‡</td>
<td>.1753</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>EntrezGene</td>
<td>.5250</td>
<td>.1751</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>UMLS+MeSH</td>
<td>.5236</td>
<td>.1746</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>UMLS+EntrezGene</td>
<td>.5254</td>
<td>.1744</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>MeSH+EntrezGene</td>
<td>.5223*</td>
<td>.1752</td>
</tr>
<tr>
<td>Acronym expansion with KBs</td>
<td>UMLS+MeSH+EntrezGene</td>
<td>.5233‡</td>
<td>.1746</td>
</tr>
<tr>
<td>Document and passage retrieval score combiner</td>
<td>Exponential of normalization</td>
<td>.5353</td>
<td>.1773</td>
</tr>
<tr>
<td>Document and passage retrieval score combiner</td>
<td>Exponential</td>
<td>.5362</td>
<td>.1759</td>
</tr>
<tr>
<td>Document and passage retrieval score combiner</td>
<td>Logarithmic</td>
<td>.5472</td>
<td>.1744</td>
</tr>
<tr>
<td>Document and passage retrieval score combiner</td>
<td>Normalization of exponential</td>
<td>.5471</td>
<td>.1730</td>
</tr>
<tr>
<td>Document and passage retrieval score combiner</td>
<td>Normalization</td>
<td>.5473</td>
<td>.1744</td>
</tr>
<tr>
<td>Document and passage retrieval score combiner</td>
<td>No transformation</td>
<td>.5471</td>
<td>.1730</td>
</tr>
<tr>
<td>Ablation of lexical variants</td>
<td></td>
<td>.4696</td>
<td>.1613</td>
</tr>
<tr>
<td>Ablation of sentence extraction</td>
<td></td>
<td>.4545***</td>
<td>.1259***</td>
</tr>
<tr>
<td>Ablation of proximity based ranking</td>
<td></td>
<td>.5444</td>
<td>.1693‡</td>
</tr>
<tr>
<td>Filtering overlapping passages</td>
<td></td>
<td>.5342</td>
<td>.1773</td>
</tr>
</tbody>
</table>
mance. However, they also contained more noisy content than MeSH, which hurt the performance when combining with MeSH. As most acronyms were expanded as synonyms, integrating any combination of KBs hardly affected the performance, though we find that EntrezGene is a useful source for acronym expansion compared with others. Unsurprisingly, we confirm that synonyms should be considered in extracting important sentences and reranking, and filtering overlapping passages could better help the task than only filtering identical passages. Different transformation strategies hardly affected the overall performance.

For the real-valued parameters, we can see from Figure 4.2 that altering weights for concept terms or verbs could greatly change the performance. In particular, concept terms favored higher weights and verbs favored moderate weights ($\sim 0.4$). Jelinek-Mercer outperformed Dirichlet smoothing for this task with the best performance achieved when the parameters were set as 0.1 and 1000. We also found the performance was improved with the parameters set to lower values than those reported in their original paper for the important sentence extraction and reranking algorithms [148].
4.3.5 Exploration strategy performance analysis

In this subsection, we demonstrate how the exploration strategy discovered the best system presented in Section 4.3.3 by analyzing the parameters estimated by the framework at different analysis steps.

We show in Figure 4.3 the likelihood and posterior distributions of cost estimated by the exploration strategy when the experiment completed. First, Figures (a) and (b) represent likelihood estimation of cost for five steps: score combination methods (C), overlapping passage solutions (O), retrieval strategists (R), sentence extraction and reranking (S), and term proximity based reranking (P) respectively. We can clearly see that the framework discovered each step had a different distribution of execution time, and C and O components usually performed faster than R, S, and P. Figures (c) to (d) represent likelihood estimation of component-level cost for these steps. We see that the configured components in each of C, O, and P had similar cost distributions, while the behavior of different S and R components varied, since the computational complexity of these components tended to be easily affected by different configurations. Figures (e) to (h) are the posterior estimations of $\gamma_t$ and $\gamma^c_t$ to predict $c(f^c_t|\omega^c_t)$ for unseen components. Since enough samples were provided at the end, the mean of each $\gamma_t$ or $\gamma^c_t$ was close to that of the likelihood estimation, and since more samples were available to estimate the step-level costs than the component-level costs, the variance of $\gamma_t$ tended to be smaller than that of $\gamma^c_t$.

We further investigate how the likelihood and posterior distributions of cost changed through the execution process by showing both distributions estimated when the experiment completed 1% of total execution tasks in Figure 4.4. Comparing with Figure 4.3, we see that the means of both distributions were smaller than those estimated at the end, due to the randomness of initially sampled components and input sets; the variances of the likelihood distributions were smaller due to fewer samples. According to Algorithm 1, components of downstream steps (C and O) tended to be sampled and tested more frequently than those of upstream steps (R, S, and P), due to smaller LCC scores; variances of the estimated posterior distributions of R, S, P components tended to be greater than those of C and O, which is clearly shown in Figures (e) to (h).

Estimating benefit had a similar process as estimating cost. We use various configurations to illustrate the estimated likelihood and posterior distributions of benefit when the 20% of total execution tasks were completed. Figures 4.5(a) and (b) show the likelihood estimation of the benefit for different configurations of the reranking algorithms and KBs, which exhibited different effects. Since KBs were integrated at the beginning at the pipeline while reranking algorithms were placed at the end, very few combinations of various KBs had been tested thus far, which correspond to the few “sharp” distributions, and the likelihood distributions corresponding to others remain unknown; on the other hand, different reranking methods were tested more equally frequently, as a result, their distributions were more alike. This assumption can also be justified from Figures (c) and (b), which correspond to the posterior estimations of both component-level benefit means, where variances of posterior distributions for reranking algorithms were close, while those corresponding to the untested KBs took the global variance.
Figure 4.3: Estimation of likelihood and posterior of cost at the end of the TREC Genomics experiment. (a) and (b) represent likelihood estimation of step-level cost, where C, O, R, S, P correspond to score combination methods, overlapping passage solutions, retrieval strategists, sentence extraction and reranking, term proximity based reranking respectively. (c) and (d) represent likelihood estimation of cost at component level. (e) to (h) are the posterior estimations of cost mean $\gamma_t$ and $\gamma_c$. X-axis is the execution time in seconds.
Figure 4.4: Estimation of likelihood and posterior of cost when the TREC Genomics experiment completed 1% of total execution tasks. (a) to (h) correspond to Figure 4.3 (a) to (h). X-axis is the execution time in seconds in all subfigures.
4.3.6 Generalizability analysis

In this subsection, we first demonstrate the performance of system selection results by various resampling methods. Besides the leave-one-out cross-validation, we also applied 5-fold cross-validation, 10-fold cross-validation (each repeated 100 times) and bootstrap (repeated 300 times) for model selection and performance evaluation. We find that the 10 highest ranked systems produced by all the resampling methods are identical; we further compare the performance of all systems evaluated on the validation set and test set in Figure 4.6(a). We can see that the systems produced by the proposed framework with a resampling based system selection method are generalizable to unseen data for the task, and cross-validation methods are less biased than bootstrap method.

We also tested these automatically selected configurations on 36 topics from TREC Genomics 2007 to test their adaptability for a different but similar task. TREC Genomics 2007 topics are in the form of questions asking for lists of specific entities, e.g. “What [GENES] are genetically linked to alcoholism?”. For this example question, the expected output would be a ranked list of passages that relate one or more entities of type GENE to alcoholism.
Figure 4.6: Generalizability of systems with various resampling methods on (a) holdout topic set from 2006 and (b) unseen topic set from 2007 in the TREC Genomics experiment. X- and Y-axis correspond to PsgMAP scores.

<table>
<thead>
<tr>
<th>Table 4.5: Result on TREC Genomics 2007 data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocMAP</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Max</td>
</tr>
<tr>
<td>Max</td>
</tr>
<tr>
<td>Participants</td>
</tr>
<tr>
<td>CSE</td>
</tr>
</tbody>
</table>

We used the 2007 topic set to test the configurations selected by the exploration strategy for the 2006 data set (Table 4.5). We can see that the best system slightly outperformed the best reported in [61]. The difference in performance on the 2006 and 2007 datasets is plotted in Figure 4.6(b), which is clearly larger than the performance difference between different folds drawn from the 2006 data set. The best configuration is also different, and favors different KBs and weights. Nevertheless, the coefficient of correlation between 2006 and 2007 tests (0.272) still confirms that the best configurations from 2006 tend to perform better for 2007. We also see that data points are clearly clustered into groups, and the performance difference between clusters tends to be more significantly different than that within each cluster. This indicates that systems which statistically significantly differ from the top systems should be explored to better diversify the system outputs and enhance the likelihood of achieving optimal performance.

4.4 General Exploration Strategy

In this section, we model the decision processes in addition to the execution processes introduces in Section 4.2.2. and then study the overall optimization solution.
4.4.1 Modeling Decision Processes

Synthesizing factor-level assertions into a comprehensive decision is non-trivial. Intuitively, not only does it need to evaluate the importance of each factor, i.e. the weight or priority in traditional decision analysis, but it is also more important to map combinations of assertions from sub-factors into assertions of their super-factor. In this section, we present a solution synthesis strategy $\sigma$ that multiplies the factors in the decision process $p$. We derive algorithms for making decisions and learning solution synthesis strategies.

We first recall from Section 2.2 that a problem reduction function $\rho$ is a function defined on $\mathbb{R}^{D(p)} \rightarrow \mathbb{R}^{D(p_1) \times \ldots \times D(p_n)}$ and a solution synthesis function $\sigma$ is a function defined on $\mathbb{R}^{T(p_1) \times \ldots \times T(p_n)} \rightarrow \mathbb{R}^{T(p)}$. In this case study, we make a probabilistic assumption about the two functions, i.e.

$$p^\rho(x_0, \ldots, x_n | x) : D(p) \times D(p_0) \times \ldots \times D(p_n) \rightarrow \mathbb{R}$$

$$p^\sigma(y | y_0, \ldots, y_n) : T(p_0) \times \ldots \times T(p_n) \times T(p) \rightarrow \mathbb{R}$$

where we use $p_0$ to denote the decision factor which replicates the original decision task but only requiring an execution process with no further decision decomposition, which can be considered as a backup solution that secures the task performance no worse than an information pipeline that involves only execution processes. We made a further assumption that the decision factors are independent in the problem reduction function, i.e. $p^\rho(x_0, \ldots, x_n | x) = p^\rho_0(x_0 | x) \ldots p^\rho_n(x_n | x)$.

Each $p^\rho_i(x_i | x)$ represents a decision maker’s preference or intuition how a subtask $x_i$ can influence the original task $x$, and $p^\sigma(y | y_0, \ldots, y_n)$ represents the mapping from the subtask outputs $y_0, \ldots, y_n$ to the original task output $y$.

We combine the modeling of the decision logic $p^\rho_0(x_0 | x), \ldots, p^\rho_n(x_n | x)$ and $p^\sigma(y | y_0, \ldots, y_n)$ with the modeling of the execution processes instantiated for the sub analytics procedures $p(y_0 | x_0), \ldots, p(y_n | x_n)$ described in Section 4.4. We can model the decision process as the joint probability as follows

$$p^\sigma(y | y_0, \ldots, y_n) \prod_i p^\rho_i(x_i | x)p(y_i | x_i)$$

4.4.2 Overall Optimization

Based on the modeling of decision processes and execution processes, we propose the overall optimization objective and describe our solution. In addition, we would like to answer many open theoretical questions and challenges that have not yet been addressed. For example:

- How hard is the optimization problem? Is it intractable? In other words, how much exploration effort can be reduced from the exhaustive search over the analytics space?
- What are the other exploration strategies? How do they compare with the proposed solution in terms of the expected performance and/or the performance for the best/worst case?
- What is the asymptotic or dynamic behavior of the proposed solution and/or other solutions? This can help estimate whether we should terminate the iterative development and improvement process at an earlier time.
• Is our assumption reasonable that the performance metrics follow a hierarchical Bayesian (or Gaussian in particular) distribution? Is there another way to capture such features?
• Can we estimate the overhead, i.e. how much extra cost is required to perform the exploration algorithm? How does it compare to the cost of executing a pipeline?

[Proposed Work, c.f. Section 4.5 and Chapter 9]

4.5 Proposed Work

We review literature on automated service composition, planning and scheduling, and attempt to design a general exploration strategy for the analytics space exploration problem, and in addition we would like to answer many open theoretical questions and challenges that have not yet been addressed.
Chapter 5
Analytics Meta Learning Implementation

This chapter focused on the development detail of the intelligent architecture layer that assists developers in building information systems using analytics meta learning methodology. In Section 5.1, we discuss the high-level design and engineering principles of the architecture layer for our implementation as well as those who intend to implement for their in-house architecture frameworks other than UIMA. We then in Section 5.2, we describe our current implementation based on Apache UIMA and uimaFIT1. Finally, in Section 5.2.1 we proposed future work.

5.1 Principles of Design and Engineering

To facilitate easy integration with existing components, we require a system that is simple to configure and powerful enough to sift through thousands to trillions of option combinations to determine which represent the best system configuration. In this section, we present the general principles in implementing such architecture frameworks for in-house system development, which include the mandatory modules: declarative descriptors, configurable evaluation, automatic data persistence, and configurable exploration strategy, as well as optional but recommended modules: global resource caching, distributed architecture, and graphical user interface.

5.1.1 Declarative Descriptors

To leverage the analytics meta learning framework, users specify how the analysis components should be organized into an information system, which values need to be specified for each configuration, what is the input benchmark set, and what measurements should be applied. As we can see from the evolution of cognitive systems in Section 2.1, only if we can describe the elements effectively, can an algorithm be applied to manipulate and reason about them.

Compared to describing the system and component in the code, such as programatically itemizing the steps of an information pipeline in a main() method or via uimaFIT, and conveniently configuring the components via Java annotations, we recommend standalone workflow representations, such as XML or YAML based process modeling languages. It is because such representation

1 https://uima.apache.org/uimafit.html
language is generally independent of implementation details (e.g. the programming language, the system requirement, etc.), and it can be more easily accessed and interpreted by the architecture framework.

5.1.2 Configurable Evaluation

Analytics meta learning supports the evaluation of component performance based on user-specified evaluation metrics and gold-standard outputs at each step; these measurements are also used to estimate the benefit and utility for each component and sub process. Analytics meta learning also requires the capability to calculate statistical significance of the performance difference between systems on a given task. This is important when attempting to disprove the null hypothesis that the best baseline system is as good as alternative systems, and crucial to understanding each component’s contribution (Section 4.3.4) and prioritizing the candidate systems when adapting to new tasks (Section 4.3).

Ideally, the framework should provide a set of commonly used evaluation metrics, such as precision, recall, F-measure. In addition, the framework should also provide APIs to allow users to extend the built-in evaluation metrics to meet various specific needs, which usually includes two abstract methods that respectively evaluate for each instance and aggregate across the entire development set. For example,

```java
Map<Measure, Double> calculate(Collection<T> resultEvaluatees, Collection<T> gsEvaluatees);
Map<Measure, Double> aggregate(
    Map<Measure, ? extends Collection<Double>> measure2values);
```

5.1.3 Automatic Data Persistence

As mentioned in Section 4.2, if two system traces share the same prefix, it isn’t necessary to repeat executions for the configured components along the prefix if we assume that intermediate results are kept in a repository accessible from any system. This also allows the system to pause, resume and continuously integrate new modules during the exploration process. Moreover, intermediate data are also useful for error analysis, system performance analysis, and reproduction of previous experimental results.

To achieve automatic data persistence, the framework should accurately and efficiently serialize arbitrary data objects output from analysis components and deserialize from the storage before sending them to the next analysis component, and the storage should be able to quickly locate the Binary Large OBject (BLOB) from the key pair of input and the trace prefix. Since we expect this process to be transparent to the module developers, i.e. they do not need to add serialization and deserialization code to each component analysis, we recommend to utilize existing serialization frameworks such as Apache Thrift\(^2\) or data processing architecture frameworks that support data serialization.

\(^2\)http://thrift.apache.org/
5.1.4 Configurable Exploration Strategy

The component selection strategies can be configured by the user and extended by the integration tester; strategies corresponding to the heuristic function described in Section 4 should be implemented.

5.1.5 Global resource caching

Online resources are sometimes temporarily unavailable, and are occasionally updated, changing their contents; to support reproducibility of results, we implemented a generic resource caching strategy inside the analytics meta learning framework. In addition to ensuring the reproducibility of results when external KBs are queried, global resource caching can also speed up the exploration process by avoiding replicated service calls across systems.

5.1.6 Distributed Architecture

In Section 4, we focus on discussion of single-machine solutions to the analytics meta learning problem, i.e. only one configured component is processing one input instance at any time. In fact, we can also make the exploration process run in parallel, where the configured components are deployed into the cluster beforehand; the execution, fault tolerance and bookkeeping are managed by a master server.

5.1.7 Graphical User Interface

A Web GUI could help the developers to easily observe the current status of the exploration process and the performance of the produced systems. We recommend three different use cases for the Web GUI: experiment inspection, ad hoc problem solving, and trend analysis.

- The experiment inspection function helps users monitor configuration exploration progress and evaluation results, and view current and past persisted intermediate data for error analysis.
- The ad hoc problem solving function enables users to submit an ad-hoc analytics task similar to those used in the benchmark set and inspect the output from the current best system.
- The trend analysis view allows users to visualize performance change across experiments over a selected period of time.

5.2 Open Source UIMA Extension

In this section, we present the analytics meta learning framework implementation, a distributed system for a parallel experimentation test bed based on UIMA, configured using declarative descriptors. We highlight the features of the implementation in this section. Source code, examples, and other resources are publicly available.
**Declarative descriptors.** Our implementation introduces *extended configuration descriptors* (ECD) based on YAML\(^3\) format. We further extend the previously implemented CSE framework to support the unified representation language for arbitrary analytics task and analytics procedure.

[Proposed Work, c.f. Section 5.2.1 and Chapter 9]

**Configurable evaluation.** Commonly-used information retrieval metrics are implemented in the open source software.

**Automatic data persistence.** Our implementation includes an automatic persistence strategy for data and experiment configuration.

**Configurable exploration strategy.** Strategies corresponding to the heuristic function described in Section 4 are implemented in the open source software.

[Proposed Work, c.f. Section 5.2.1 and Chapter 9]

**Procedural knowledge discovery module.** We integrate the automatic analytics procedure discovery modules from Chapter 3.

[Proposed Work, c.f. Section 5.2.1 and Chapter 9]

### 5.2.1 Proposed Work

We first integrate the automatic analytics procedure discovery modules from Chapter 3 and implement additional exploration strategies from Chapter 4. We then extend the previously implemented CSE framework to support the unified representation language for arbitrary analytics task and analytics procedure.

\(^3\)http://www.yaml.org/
Chapter 6

Case Study: Biomedical Question Answering Task

In the next three chapters, we focus on exemplifying the proposed methodology to tackle analytics tasks and reporting the performance of the proposed analytics meta learning method and its resulting products. We first describe the task or the task family of each chapter. Then, we will follow the general solution recipe proposed in Section 2.3 to solve each individual analytics task, and in each respective section, we give our manual definition of the analytics procedure or describe the automatic analytics procedure definition process, and present analysis component integration or development, and analytics space exploration, which is followed by result reports. This chapter presents the use of analytics meta learning in construction of an information system that supports a series of real-life information seeking needs from biomedical experts, e.g. the Task Examples 1 to 5 in Chapter 1.

We first review prior biomedical information systems in Section 6.1. In this case study, we manually define the analytics procedure following a general question answering pipeline [151]. We describe the task in detail in Section 6.2. In Section 6.3, we adapted a leveraged a UIMA-based three-layered architecture that was previously developed for biomedical QA tasks including TREC Genomics questions [169], CLEF QA4MRE questions [118], and BioASQ challenge questions [171]; the architecture consists of a generic component layer (BaseQA), a biomedical component layer (BioQA) and a BioASQ-specific component layer. In Section 6.4, we present the analysis components that we have developed. Using the development set, we also investigated whether it is possible to design and train supervised models to answer factoid and list questions, without the use of manually-constructed rules or predefined templates. We utilized supervised models to merge answer scores obtained from various sources, a technique utilized by some systems in past years [115, 163], and to predict likely answer type(s) from among the 133 semantic types in the UMLS Semantic Network. We report our preliminary results in Section 6.5. In Section 6.6, we analyze the results from the analytics space exploration process. In Section 6.7, we detail the development plan for the missing functionalities, such as yes/no QA, summary generation, etc., to complement the current system, as well as the experiment plan for analysis component and execution process.

[https://uima.apache.org/]
optimization using the proposed exploration strategies.

The system developed using the analytics meta learning methodology has participated in the BioASQ QA challenge. Our hypothesis is that informatics challenges like BioASQ are best met through careful design of a flexible and extensible architecture, coupled with continuous, incremental experimentation and optimization over various combinations of existing state-of-the-art components, rather than relying on a single “magic” component or single component combination. The architecture frameworks and most of the components are currently available as open-source downloads, and we are planning to release the remaining components that are used in the system as open source software in the near future.

6.1 Related Work

We focus on summarizing systems and approaches that have participated in shared tasks related to biomedical question answering, e.g. TREC Genomics [59] and BioASQ [154], or have leveraged the task benchmarks outside the formal competition.

A typical biomedical question answering pipeline consists of three major analysis components: keyterm extraction and query expansion, document (or paragraph) retrieval, and passage extraction [131, 140, 148]. Synonyms, acronyms, and lexical variants are processed in the first phase; a retrieval model is applied in the second phase; and the similarity between the query and each retrieved passage is considered in the last phase. Researchers have tried to apply existing general retrieval systems [14, 132], relevance feedback to the traditional retrieval model [93, 143, 173], or a fusion or shrinkage of retrieval models [55]. Moreover, linguistic knowledge is also incorporated for the task, e.g., POS tags [100], and SVO structure [148].

Recently, researchers in biomedical information retrieval and question answering continue to leverage the TREC Genomics data set for evaluation [22, 64, 90, 101, 108, 153, 174]. Given the absence of an easy-to-use framework for building baseline systems for new tasks and exploring large parts of the configuration space, iterative research typically focuses on perturbations to a single module while keeping modules and parameters elsewhere in the system frozen. Stokes et al [147] follow the same assumption to combine components from task participants to find optimal configurations for a biomedical IR task, using an approach specific to the task and difficult to generalize.

We further survey more recent BioASQ systems.

[PROPOSED WORK, C.F. SECTION 6.7 AND CHAPTER 9]

6.2 Task Description

A biomedical question answering task defines infinite concrete analytics tasks, which can be defined by pairs of information seeking needs described in natural language (e.g. Task Examples 1 – 5 in Chapter 1) and expected exact answers (e.g., named entities in the case of factoid and list questions) and ideal answers (paragraph-sized summaries) [95]. For example, Task 1

How many TAp73 isoforms have been identified in humans?
should have an exact answer of either
    seven or 7

and an ideal answer of
    The TP73 gene, due to the presence of two promoters (P1 and P2) in its 5’ flanking region, encodes a fully transcriptionally active domain (TAp73) and the amino terminus deleted (∆Np73). TAp73 possesses pro-apoptotic properties, while deltaNp73 has anti-apoptotic functions. Alternative 3’-end splicing results in generation of at least seven TAp73 distinctive isoforms (α, β, γ, etc).

In addition, the user would be interested in knowing the evidence that supports the answer, including relevant PubMed articles (document)

PMID: 18256531
PMID: 22388545
PMID: 23159862
...

relevant text snippets within the documents

A member of the p53 family, p73, has several isoforms and differentially regulates transcription of genes involved in the control of the cell cycle and apoptosis. (0 - 162 from the abstract of PMID: 18256531)
The expression of all 5 N-terminal isoforms (TAp73, DeltaNp73, DeltaN’p73, Ex2p73 and Ex2/3p73) was measured by real-time RT-PCR and p53 status was analyzed by immunohistochemistry. TAp73, DeltaNp73 and DeltaN’p73 were significantly upregulated in tumors. (347 - 602 from the abstract of PMID: 16964385)
...

and relevant concepts from knowledge bases

MeSH Heading: Protein Isoforms (D020033)
...

6.2.1 Evaluation metrics

As motivated in Chapter 1, any domain expert can hardly solve all the analytics tasks, not to mention an information system. In order to automate the system construction process, we need to
define metrics to estimate to what extent the system is able to answer the questions. We adopt the evaluation metrics that have been adapted for similar scenarios to measure different types of output [59][154].

The relevance of documents and concepts is measured by precision@N, recall@N, F-measure@N, average precision@N (AP@N) at the task/instance level and averaged at the experiment level, where are referred to as mean precision@N, mean recall@N, mean F-measure@N, mean average precision@N (MAP@N). In particular, they are defined as follows:

\[
\text{precision}@N = \frac{\text{true positive}@N}{\text{true positive}@N + \text{false positive}@N}
\]

\[
\text{recall}@N = \frac{\text{true positive}@N}{\text{true positive}@N + \text{false negative}@N}
\]

\[
\text{F-1}@N = 2 \cdot \frac{\text{precision}@N \cdot \text{recall}@N}{\text{precision}@N + \text{recall}@N}
\]

\[
\text{AP}@N = \frac{\sum_{i=1}^{N} \text{precision}@N \cdot \text{rel}@N}{\min(|\text{relevant items}|, N)}
\]

where rel@N equals 1 if the N-th returned item is relevant and 0 otherwise. The relevance of snippet (or passage) is measured by Passage MAP and Passage2 MAP [59], which consider the character level overlapping between the system generated results and the gold standard outputs.

The correctness of the system generated exact answers are estimated using accuracy@N, reciprocal rank@N, precision, recall, F-measure, depending on the task (question) type, at the task/instance level and averaged to mean accuracy, mean reciprocal rank (MRR), mean precision, mean recall, mean F-measure, and reduced to strict accuracy and lenient accuracy for experiment level. In particular,

- if the question or task description requires either a “yes” or “no” as an answer (a.k.a yes/no question), then the task performance is judged using binary accuracy@1, i.e. 1 if correct and 0 if wrong at the task/instance level, and mean accuracy at the experiment level.
- if the question or task description requires a particular entity (e.g. a disease, drug, or gene) as an answer (a.k.a factoid question), then the task performance is judged using accuracy@1 (strict accuracy), accuracy@5 (lenient accuracy), i.e. 1 if there exists a correct answer in the first five answer candidates, and reciprocal rank@N, i.e. the inverse of the rank of the first correct answer in the candidate list.
- if the question or task description requires a list of entities (e.g. a list of genes) as an answer (a.k.a list question), then the task performance is judged using precision, recall, and F-measure.

If the question or task description can be responded only by a short text summarizing the most prominent relevant information, we call it summary question or non-factoid question. To judge the performance for ideal answers for yes/no, factoid, list, and summary questions, we use ROUGE-2 and ROUGE-SU4 [85], which counts the overlapping N-grams between the system
generated summary and the reference or golden summary.

\[
\text{ROUGE-}N = \frac{\text{overlapping } N\text{-grams}}{\text{N-grams in golden summary}}
\]

\[
\text{ROUGE-SU}N = \frac{\text{overlapping skip bigrams limited to } N + \text{overlapping unigrams}}{\text{skip bigrams in golden summary limited to } N + \text{unigrams}}
\]

6.2.2 Dataset

Besides evaluation metrics, another crucial element when using analytics meta learning for constructing information systems is a benchmark dataset. In fact, a number of shared tasks have been organized to evaluate the performance of biomedical question answering (QA) systems. For example, TREC Genomics QA [59] focused on genomics questions for a subdomain of biomedical research, and made available a relatively small set of factoid and list questions (a total of 64 questions from two years). Recently, the CLEF QA4MRE task [120] organized a pilot track on multiple choice QA for questions related to Alzheimer’s disease. Compared to these shared tasks, the BioASQ challenge [154] covers a wider range of biomedical subdomains, and releases a larger topic set with much more detailed gold standard data, including relevant documents, snippets, concepts, triples and both exact and ideal answers. We compare them in Table 6.1.

We have demonstrated using TREC Genomics benchmark dataset for biomedical passage retrieval system in Section 4.3, we use the BioASQ QA benchmark set for optimizing the biomedical question answering system.

6.3 Analytics Procedure Definition

In this case study, we consider two fixed information processing pipeline – Phase A and Phase B pipelines. Phase A pipeline retrieves relevant concepts, triples, documents, and snippets, and Phase B pipeline generates answers for specific types of questions. We utilize the conventional information retrieval and question answering methodology to design both pipelines, and we illustrate them in Figure 6.2 for Phase B and Figure 6.1 for Phase A.

To integrate the analysis components into the analytics procedures at each analysis step and facilitate the extensibility of the system, we design a three-layered architecture uses the analytics meta learning framework.

The first layer BaseQA\(^2\) is designed for domain-independent analysis components, and includes the basic input/output definition of analytics engine, intermediate data objects (such as answer type, question type, relevant passages, relevant concepts, etc.), analytics task evaluation components, and data processing components (e.g. LingPipe\(^3\) and Apache OpenNLP\(^4\) wrappers, Lucene\(^5\)-based passage retrieval component, LibLinear\(^6\) wrapper, and models applicable to

\(\text{https://github.com/oaqa/baseqa/}\)

\(\text{http://alias-i.com/lingpipe/index.html}\)

\(\text{https://opennlp.apache.org/}\)

\(\text{https://Lucene.apache.org/}\)

\(\text{http://www.csie.ntu.edu.tw/~cjlin/liblinear/}\)
Figure 6.1: BioASQ QA Phase A pipeline diagram. * represents a provider that requires accessing BioASQ Web services.
Figure 6.2: BioASQ QA Phase B pipeline diagram. † represents a provider that requires accessing external Web services.
<table>
<thead>
<tr>
<th>Example</th>
<th>TREC Genomics</th>
<th>BioASQ QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1</td>
<td>What is the role of PrnP in mad cow disease?</td>
<td>Is TREM2 associated with Alzheimer’s disease?</td>
</tr>
<tr>
<td>Example 2</td>
<td>What [GENES] regulate puberty in humans?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coverage</th>
<th>genomics</th>
<th>biomedical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years</td>
<td>2006, 2007</td>
<td>2012 – present</td>
</tr>
<tr>
<td>Benchmark size</td>
<td>64</td>
<td>810 (as of 2014)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1307 (as of 2015)</td>
</tr>
</tbody>
</table>

| Question type | summary, factoid | factoid, list, yes/no, summary |
| Answer type   | relevant documents and passages | relevant documents, snippets, concepts, triples, ideal answers, exact answers |

| Best reported results | Document MAP: ~0.50 | Document MAP: 0.26 – 0.30 |
|                       | Passage MAP: ~0.17  | Passage MAP: 0.04 – 0.06 |
|                       |                      | Concept MAP: 0.43 – 0.68 |
|                       |                      | Triple MAP: 0.04 – 0.09 |
|                       |                      | Answer MRR: 0.05 – 0.28 |
|                       |                      | (as of 2014, scores vary across different test batches) |

| Corpus | PMC Open Access full texts | PubMed abstracts, PMC Open Access full texts |

This table shows a comparison of biomedical question answering shared tasks: TREC Genomics and BioASQ. Although this case study focuses on the biomedical domain, it is the first similar shared task to the best of our knowledge, to combine four types of questions and evaluate both exact and ideal answers along with other relevant elements (e.g., triples), so many aspects of the existing BaseQA framework were extended to accommodate BioASQ application development. We modified the intermediate object and input/output object definition (UIMA type system) according to the task requirements. For example, we added two new attributes `Begin/EndSection` to each `Passage` type, and changed the `Begin/EndPosition` attributes to `Begin/EndPositionInSection`. We also provided a BioASQ-compatible JSON format reader and writer at the BaseQA level, which we believe can be widely used in various analytics tasks beyond BioASQ. We also implemented the general-purpose evaluation methods at BaseQA layer.

In the second layer (BioQA), we implemented biomedical resources that can be used in any biomedical analytics task (outside the context of BioASQ), including UMLS Terminology Services (UTS)\(^7\)-based synonym expansion component, a MetaMap annotation component, etc. For the components that are included in the BaseQA layer, we also created a descriptor for the component

\(^7\)https://uts.nlm.nih.gov/home.html

86
at the BioQA level by overriding the model value with a path to the specific model tailored for biomedical domain, where applicable. For example, the ClearNLP wrapper, which is provided at the BaseQA level with the default general-en model specified in the descriptor, has a new descriptor for the bioinformatics-en model, trained on the CRAFT treebank [160], defined at the BioQA level. Although the training and testing processes are performed on the BioASQ development set, the derived models can also be used for other biomedical questions, so we also place the models and training components in the BioQA layer.

A few BioASQ evaluation specific components were integrated in the third design layer; for example, GoPubMed services are only hosted for the purpose of the BioASQ challenge. The introduction of this task-specific layer will facilitate easy replacement of proprietary and restricted components when we adapt the system to other biomedical analytics tasks or deploy the system as a real-world application. The end-to-end training and testing pipelines are also defined in this layer. Similar to the resource-wrapper providers which we introduced for the TREC Genomics task [169], we also created a caching layer, using Redis [8] for all outgoing GoPubMed service requests, along with a Java client for accessing either the official GoPubMed server or the caching server, specified by a properties file [9] which helps to reduce the workload of the official server and reduce experiment run-time when multiple developers are evaluating their components.

6.4 Analysis Component Construction: Learning to Answer Biomedical Questions

In this section, we detail the analysis components that have been integrated for analytics meta learning. As we present in Section 6.3, analysis components in Phase A do not distinguish types of input task descriptions. We first focus on analysis components at the following analysis steps in Phase A, including the document retrieval component in Section 6.4.1, the snippet retrieval component in Section 6.4.2, the concept retrieval component in Section 6.4.3, and the triple retrieval component in Section 6.4.4. Factoid and list QA tasks have similar topic distributions and linguistic structures, and each exact answer also uses a similar language representation. Accordingly we designed two supervised models that are shared by both question types for Phase B: answer type prediction component in Section 6.4.5 and candidate answer scoring component in Section 6.4.7, which allows us to best leverage the training data. In addition, we introduce approaches for candidate answer generation in Section 6.4.6. In comparison to factoid questions, list questions require the system to return a list of exact answers, of the same type, which requires an answer pruning component for list question answering only, which is described in Section 6.4.8.

6.4.1 Document Retrieval

Our approach is similar to Xu and Callan proposed for the TREC 2014 Web Track [166], with some modifications made for better performance and efficiency.

http://redis.io/
https://github.com/zly/bioasq-gopubmed-client/
Offline Indexing of Medline Baseline Corpus. We used Lucene to index a Medline baseline corpus using title, abstract and keywords fields, if available. We used the standard Lucene tokenizer combined with the Krovetz Stemmer, which is less aggressive compared to the Porter Stemmer. This is an important step, because many biomedical terms (in particular gene names) are not recognizable by stemmers, and the Porter stemmer is likely to truncate many of the words, causing increased confusion between the stemmed biomedical terms and common terms during search time. We also kept the stopwords in the index. The motivation is that since we only have the abstract text for the document, removing stopwords may result in less accurate field length statistics, thus affecting the performance of many language model based retrieval models.

Hierarchical Retrieval Architecture. The fact is that given a query, we have more retrievable documents than we can perform a deeper analysis for. However, to ensure better retrieval performance, in-depth analysis of the documents is necessary. Therefore, a hierarchical retrieval architecture is introduced here to find a good balance between performance and efficiency. In summary, each search task is processed by three stages:

1. Obtaining an affordable high recall candidate set. During the query time, we have removed all stopwords from the query, as they provide no useful information and will likely cause serious efficiency issues. We use the Dirichlet smoothing retrieval model implemented in Lucene to conduct this search. In our implementation, we consider only the top 10,000 ranked documents.

2. Precision oriented reranking. We incorporate the Negative Query Generation (NQG) model [94], which utilizes a negative language model by assuming that all documents in the corpus are non-relevant, thus making more accurate adjustments to query term weights and relevance calculations. After re-ranking with NQG, we can now further cut down the candidate set by considering only the top 100 documents in the ranked list.

3. Deep document feature extraction and Learning TO Rank (LETOR). We use ranker scores (e.g. BM25, Jelinek-Mercer smoothing, Dirichlet smoothing, Indri two-stage smoothing, NQG, etc), similarity scores (e.g. Jaccard coefficient and Dice coefficient, etc.), raw features (e.g. document length, vocabulary size, etc.), and customized features (e.g. harmonic means of the ranker scores across all fields, the distribution of the query terms across the documents, etc.). We simply score the $K$ documents with a pretrained LETOR model which was optimized for Precision@10. Here, we are using Random Forest, an ensemble method known for robustness against overfitting.

The details of the proposed document retrieval approach can be found in [166]. However, we intend to further improve the document retrieval performance.

6.4.2 Snippet Retrieval

The snippet retrieval module analyzes the 10 most relevant documents returned from the upstream document retrieval component. We first identify the extent of a snippet and then apply a LETOR approach for snippet retrieval.
Table 6.2: BioASQ Phase B Snippet Retrieval Features

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BM25: We index all the candidate snippets using Lucene, and then use a query that contains not only words but also phrases and confidence scores of all the different query concepts returned by the MetaMap service.</td>
</tr>
<tr>
<td>2</td>
<td>Skip-bigram: Based on the dependency relations generated from the dependency parser for each question, we count the number of matched pairs and calculate the F-1 based skip-bigram score.</td>
</tr>
<tr>
<td>3</td>
<td>Textual alignment: Surface similarity of a snippet and a question. We also consider the relative order of the different words.</td>
</tr>
<tr>
<td>4</td>
<td>Some other question independent features, such as the length of the snippet.</td>
</tr>
</tbody>
</table>

**Candidate Snippets Generation.** The definition of “snippet” is the original piece of text extracted from the document. In our initial study, we found that the distribution of snippet length in the gold standard answers is similar to that of sentence length. Therefore, we apply a sentence segmenter to split the snippets and define each sentence as a snippet candidate.

**Feature Extraction and LETOR.** We define four types of features for LETOR in Table 6.2, and also apply the logistic regression classifier for scoring.

We intend to further improve the snippet retrieval performance.

[PROPOSED WORK, c.f. SECTION 6.7 AND CHAPTER 9]

### 6.4.3 Concept Retrieval

We first identify the text spans from each question and search these texts from various GoPubMed concept services. Since only a single list of concepts is returned, we also propose to merge and rank the concept lists returned from multiple sources.

**Candidate Queryable Concept Generation.** We use MetaMap to identify the UMLS concepts from the question, and our results indicate a significant improvement in recall. However, one of the major drawbacks of MetaMap is that it is poor at identifying gene and protein names. To overcome this issue, we use LingPipe NER with the model trained on the GeneTag corpus to recognize gene names to enrich the retrieved metathesaurus concepts. We then use the combination of tokens retrieved from the MetaMap service and the LingPipe NER to query various biomedical ontologies.

**Concept Ranking and Merging.** We create a ranking model that can rank the search results from different ontologies. We use the federated search approach [139], which trains a relevance mapping logistic function that maps the relevance scores of each result from each ontology to a global relevance scale.

We intend to further improve the concept retrieval performance.

[PROPOSED WORK, c.f. SECTION 6.7 AND CHAPTER 9]
6.4.4 Triple Retrieval

Similar to concept retrieval, we rely on the BioASQ provided service to retrieve relevant triples. Therefore, our goal is to construct an effective query string. Beyond the baseline method that simply concatenates all the keywords from the concept retrieval result, we made three improvements:

- Append “[obj]” and “[sub]” identifiers to each keyword in the query string.
- Enumerate all letter case possibilities for keywords: lower case, upper case, and capitalized word.
- Add all words in the original question to the keyword set while excluding the stop words and SQL reserved keywords.

The first improvement is to help the triple query server understand that most of our keywords are used as objects or subjects. This finding is intuitive through observation; since most of the words are nouns or adjectives, which are unlikely used as predicates in triples. The second improvement is based on an observation from examination of gold standard answers, where triple results indicate case-sensitivity during triple matching. Therefore, we need to include all casing variants to ensure that keywords are matched during triple retrieval. The third improvement ensures that we do not omit keywords from the original question, to make the query more robust.

6.4.5 Question and Answer Type Prediction

Previous work has studied how to define rules to extract a Lexical Answer Type (or LAT) from questions to predict the answer type, e.g. IBM’s Watson system [79]. Classification based approaches have been proposed to predict answer type from the question using syntactic and/or semantic features. Preparation of training data involves defining an answer type taxonomy manually or by leveraging existing ontologies (e.g. MUC), collecting training questions (e.g. TREC QA question set), and annotating gold standard answer type(s) [83, 112]. Weissenborn et al. [163] also define patterns for LAT extraction for BioASQ questions, and leverage the UMLS Semantic Network and map the LATs to the ontological hierarchy to obtain one of the UMLS semantic types as an “expected answer type”, which is used for type coercion checking. We took advantage of these ideas and further incorporated the Q/A pairs in the BioASQ data set for training a multi-class answer type classifier that predicts candidate answer type(s).

**Answer Type Definition.** We introduce two additional question types: CHOICE and QUANTITY in addition to the UMLS semantic types. CHOICE questions are those that have candidate answers expressed explicitly in the question, e.g.

Is Rheumatoid Arthritis more common in men or women?

We treat CHOICE questions as a special case because the candidate answers can be directly extracted from the question, and no further answer type prediction is needed. Since there exist an unlimited number of quantitative values which cannot be all covered in the UMLS semantic network, we add the QUANTITY type to complement the existing qnco (Quantitative Concept) type.

**Answer Type Extraction.** To identify the gold standard labels for the existing Q/A pairs used for training, we apply UTS to retrieve the semantic types for each gold standard exact answer, where we first use the exact search type, and if no results are returned, we further relax the
search type to words. Since UTS may return more than one concept type for each input concept, and each training question may contain more than one gold standard answer variant (these may be synonyms or answer concepts for list questions), the gold standard answer type is assigned as the most frequent concept type. If multiple concept types have the same number of occurrences for all gold standard answer variants, we keep all of them as the gold standard labels for the question.

For example, among the 44 gold standard answer variants provided for the question

In which proteins is the chromodomain present?

we can retrieve the following semantic types from UTS:

- {name: Gene or Genome, id: T028, abbr: gngm, count: 19}
- {name: Amino Acid, Peptide, or Protein, id: T116, abbr: aapp, count: 10}
- {name: Biologically Active Substance, id: T123, abbr: bacs, count: 9}
- {name: Nucleic Acid, Nucleoside, or Nucleotide, id: T114, abbr: nnon, count: 1}
- {name: Cell Component, id: T026, abbr: celc, count: 1}
- {name: Quantitative Concept, id: T081, abbr: qnco, count: 1}
- {name: Disease or Syndrome, id: T047, abbr: dsyn, count: 1}
- {name: Molecular Function, id: T044, abbr: moft, count: 1}
- {name: Food, id: T168, abbr: food, count: 1}

We assign gngm as the answer type for the question.

We identified 82 out of the 406 questions which do not have a single gold standard answer variant for which UTS can provide a semantic type. There are three major reasons for this phenomenon. First, some answer concepts are not included in the UMLS semantic network. For example, the question

Which histone marks are deposited by Set7?

has two answers:

H4K20 monomethylation

and

H3K4 monomethylation

both of which cannot be mapped to a single semantic type. However, since there are questions which ask about “which histone marks” in the training set and include gold standard answers that can be mapped to a concept type, we can simply exclude the question from the training set. Second, some gold standard exact answers do not strictly follow the representation format. For example, the question

Which enzyme is deficient in Krabbe disease?

has a gold standard answer
Galactocerebrosidase is an enzyme that is deficient in Krabbe disease (also known as globoid-cell leukodystrophy). This leads to accumulation of psychosine (galactosylsphingosine) primarily in oligodendrocytes.

In fact, “Galactocerebrosidase” alone should be the gold standard exact answer. Third, some questions, e.g.

Which is the most important prognosis sub-classification in Chronic Lymphocytic Leukemia?

with a gold standard answer

The mutational status of the IGHV genes.

have an answer which is not a simple biomedical entity, and thus cannot be mapped to a single concept type. For the second two types of phenomena, we must further investigate the linguistic and semantic structure of the answers. Finally, we obtained gold standard labels for the 324 remaining questions.

**Feature Extraction.** We first apply the ClearNLP parser to annotate the tokens, part of speech tags, and dependency relations for the question (corresponding to Lines 21 – 26 of Listing A.1 in the Section A.1). We use three approaches to identify the concept mentions in the question. We first use the MetaMap service to identify the concepts and use UTS to retrieve variant names for each concept (Lines 27 – 29). Only the first concept mapping with the confidence score returned from the service is used for each question. We also use a statistics-based LingPipe named entity recognizer (NER) (Lines 30 – 32), where the label of the named entity that is assigned by LingPipe NER is used as the semantic type of the concept. We then consider all noun phrases in the question as candidate concepts. Therefore, we employ the OpenNLP chunker to detect all noun phrases (NPs) and prepositional phrases (PPs) from each question, and extract all NPs and all NP-PP-NP occurrences (Lines 33 – 38). We then extract a number of linguistic and semantic features from the tokens and concepts, as detailed in Table 6.3. Motivated by the common structure of choice questions, we include the second and third groups of features. Since two questions that contain the same concept mentions may still have a different question focus, if the concept mentions have different syntactic roles (subject, objective, modifier, etc.) in the sentence we also add the sixth and seventh feature groups. The last feature models a simple rule-based LAT extraction approach by identifying the key noun in the sentence (Lines 39 – 41).

**Classification.** We use Logistic Regression from the LibLinear tool [40] to train a multi-class classifier, and use 10-fold cross prediction to predict a list of up to five most likely semantic labels for each question in the training set, which is used in the downstream training process (Lines 42 – 44). The model can correctly identify answer types for most high-frequency sentence patterns, such as “which genes”, but it may fail for low-frequency question patterns, where UTS may not be able to resolve ambiguous cases (e.g. AUS is identified as a country name without the context).

92
Table 6.3: BioASQ Phase B Answer Type Prediction Features

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the lemma form of each token</td>
</tr>
<tr>
<td>2</td>
<td>if the question begins with “do” or “be”</td>
</tr>
<tr>
<td>3</td>
<td>if the question contains a token “or”</td>
</tr>
<tr>
<td>4</td>
<td>if the question contains a quantity question phrase</td>
</tr>
<tr>
<td>5</td>
<td>the semantic type of each concept</td>
</tr>
<tr>
<td>6</td>
<td>a ( \langle \text{semantic type}, \text{dependency label} \rangle ) pair, where we use the dependency label of the head token in the concept bearing phrase as the second element</td>
</tr>
<tr>
<td>7</td>
<td>also a ( \langle \text{semantic type}, \text{dependency label} \rangle ) pair, where we use the dependency label of the head of the head token in the concept bearing phrase as the second element</td>
</tr>
<tr>
<td>8</td>
<td>the lemma form of the first child of the root in the parse tree that is a noun and has a dependency relation of \text{dep}</td>
</tr>
</tbody>
</table>

### 6.4.6 Candidate Answer Generation

We first use the same set of token and concept identification tools used for the question (described in Section 6.4.5) to annotate all the relevant snippets provided as input for Phase B (corresponding to Lines 51 – 65 of Listing A.1). We then integrate four components to generate candidate answers (corresponding to Lines 69 – 71, and the component level descriptor is presented in Listing A.2).

**Concepts as Candidate Answers.** We create a candidate answer using each concept identified by one of three concept identification approaches described in Sect. 6.4.5 (corresponding to Line 6 of Listing A.2). In Batch 3, we also filtered out any concept mention that is exactly a stopword, a token or phrase in the question, or a concept that is also annotated in the question. We used a stopword list that combines the most 5,000 frequent English words and the list of Entrez (PubMed) stopwords.

**CHOICE Questions** (Line 4). We first identify the “or” token in the question, and then identify its head token, which is most likely the *first option* in the list of candidate answers. Next, we find all the children of the first option token in the parse tree that have a dependency relation of \text{conj}, which are considered to be *alternative options*. We see this approach works well on most CHOICE questions, but still has problems in a few special cases. First, if two options have different prefixes but the same suffix, the suffix may be discarded in the first option, e.g.

\[
\text{Is the long non-coding RNA malat-1 up or downregulated in cancer?}
\]

We believe richer semantic interpretation of the question is needed to handle such cases. Another issue is that the head tokens can be semantically incomplete, such that a phrase which covers the head token should be used instead for the options; we expand the candidate answer using a minimal concept mention that covers the candidate answer occurrence (Line 7).

**QUANTITY Questions.** We identify all the tokens that have a POS tag of \text{CD} in all relevant snippets (Line 5). This approach can reliably produce a complete set of quantitative mentions.
However, it does not give us a way to semantically interpret the extracted numbers. For example, it could correctly identify

\[20,687,24,500, \text{etc.}\]

as candidate numbers, but does not have the ability to “summarize” the numbers and produce a single answer, e.g.

\[\text{Between 20,000 and 25,000}\]

as required. Similar to \text{CHOICE} questions, another limitation is that this method can only identify a single token as a candidate answer, e.g.

\[3.0\]

where semantically complete phrase, e.g.

\[3.0 \text{ mm}\]

is preferred. We apply the same approach used for \text{CHOICE} questions to include the \text{CD-bearing} phrase as the candidate answer (Line 7).

We intend to further improve the candidate answer generation for specific types of questions using PubTator instead of the general MetaMap tool.

\[\text{[PROPOSED WORK, c.f. Section 6.7 and Chapter 9]}\]

### 6.4.7 Candidate Answer Scoring

We predict a confidence score for each candidate answer (corresponding to Lines 75 – 77 of Listing A.1, and the component level descriptor is presented in Listing A.3). In Batch 3, we use a simple multiplication method to combine the type coercion score and the occurrence count.

In Batches 4 & 5, we define a feature space containing 11 groups of features, as shown in Table 6.4, which extend the approach used by Weissenborn et al. [163], and use Logistic Regression to learn the scoring function. We only use the questions with non-zero recall for training, where we assign “1” to each candidate answer variant if it is also contained in the gold standard answer set, and “0” otherwise. Since there are many more negative instances than positive instances, we assign to each negative instance a weight of

\[
\frac{\# \text{positive instances}}{\# \text{negative instances}}.
\]

### 6.4.8 Answer Pruning

In Batch 3, we used the factoid QA pipeline to produce answers for list questions without any pruning. In Batch 4, we used an absolute threshold to select only the answers that have a confidence score, predicted by the candidate answer scoring model, above threshold. Starting from Batch 5, instead of an absolute threshold for all questions, we use a relative threshold to filter the answers that have a confidence score above a percentage of the highest predicted score for the question (corresponding to Line 78 – 80 of Listing A.1). We tune the threshold on the development set.
<table>
<thead>
<tr>
<th>Line</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Type coercion. For each Candidate Answer Occurrence (CAO), the percentage of semantic types that are also among the top-$k$ ($k = 1, 3, \text{ and } 5$) predicted answer types. To accumulate the scores from multiple CAOs, we use “average”, “maximum”, “minimum”, “non-zero ratio”, “one ratio”, and “boolean or”.</td>
</tr>
<tr>
<td>6</td>
<td>CAO count. We use the number of CAOs for each answer variant and we also count the total number of tokens in all occurrences.</td>
</tr>
<tr>
<td>7</td>
<td>Name count. The number of distinct candidate answer names, which differs from CAO count; if two CAOs have the same text string, only one will count.</td>
</tr>
<tr>
<td>8</td>
<td>Avg. covered token count. Averaged number of tokens in each CAO.</td>
</tr>
<tr>
<td>9</td>
<td>Stopword count. For each CAO, we calculate the stop word percentage. We use the same stoplist as described in Section 6.4.6. We accumulate the scores from multiple CAOs using “average”, “minimum”, “one ratio”, and “boolean or”.</td>
</tr>
<tr>
<td>10</td>
<td>Token overlap count. For each CAO, we calculate the percentage of tokens that overlap with the question. We accumulate the scores from multiple CAOs using “average”, “non-zero ratio”, and “boolean or”.</td>
</tr>
<tr>
<td>11</td>
<td>Concept overlap count. For each CAO, we calculate the percentage of covered concept mentions that overlap with the question. We accumulate the scores from multiple CAOs using “average”, “non-zero ratio”, and “boolean or”.</td>
</tr>
<tr>
<td>12</td>
<td>Token proximity. For each CAO, we calculate the averaged distance to the nearest occurrence of each question word in the relevant snippet. We set a window size of 10, and if any question word falls out of the window, we use a fixed distance of 20. We also transform the distance to its negation and inverse, and accumulate the scores from multiple CAOs using “average”, “maximum”, “minimum”, and “non-zero ratio”.</td>
</tr>
<tr>
<td>13</td>
<td>Concept proximity. Similar to token proximity, we calculate the distance from each CAO to each question concept mention in the relevant snippet.</td>
</tr>
<tr>
<td>14</td>
<td>LAT count. For each CAO, we calculate the percentage of tokens that overlap with a LAT token (i.e. the 8th feature in Table 6.3). We accumulate the scores from multiple CAOs using “average” and “non-zero ratio”.</td>
</tr>
<tr>
<td>15</td>
<td>Parse proximity. Similar to token proximity, we use the distance in the parse tree, which is important for list questions, as answer bearing sentences may be in the form of “includes A, B, C, . . . ”.</td>
</tr>
</tbody>
</table>
6.4.9 Yes/No Evidencing

As generating candidate answers for yes/no questions are trivial, we implement the yes/no QA evidencing component according to [71].

[PROPOSED WORK, c.f. SECTION 6.7 AND CHAPTER 9]

6.4.10 Ideal Answer Generation

We implement a few idea answer generation components using sentence selection, fusion, compression, etc. techniques.

[PROPOSED WORK, c.f. SECTION 5.2.1 AND CHAPTER 9]

6.5 Preliminary Results & Analysis

Before we complete all the required analysis components for all the functionalities and apply exploration strategy to optimize the information system, we have manually improved a single information system for the analytics task. We participated in BioASQ 3B evaluation. Among all the systems that participated in Phase A evaluation, the performance of our document retrieval pipeline is scored at the bottom of the first tier. The absolute performance gaps between our pipeline and the system that is scored one place behind ours in Batches 3, 4, and 5 are measured as .0915, .0225, and .0869 respectively in terms of MAP, which are larger than those between our pipeline and the best performing system (.0435, .0204, and .0466 respectively). Over the 70 days of intensive development between April 2 to Jun 10 2015, our experiment database has recorded 717 experiments. Among 669 successful experiments, there were 167 executing the training pipeline (177.5 topics per run on average), 422 executing the testing pipeline (24.1 topics per run on average) and 80 “dummy” runs used to cache service results (284.5 topics per run on average). The official evaluation results indicate that the system achieves MRR scores of .1615, .5155, and .2727 for factoid questions, and F-measure score of .0969, .3168, and .1875 for list questions; five of these results are the highest scores reported among all participating systems.

We summarize the official evaluation results of document and snippet retrieval in Phase A and factoid and list QA in Phase B in Batches 3, 4, and 5 from the official evaluation portal in Table 6.5. The test descriptor used for Batch 5 in Phase B is shown in Listings A.1 to A.3.

Due to a relatively steep learning curve for the developers who have not had much experience with the system and the task, Phase A system used a different question analysis pipeline from the Phase B system, which had no concept retrieval module integrated and tested, which should expand each concept with synonyms. Therefore, we believe document and snippet retrieval evaluated in Phase A can be further improved by considering synonyms expanded using UTS during query formulation. Moreover, the snippets extracted by the latter snippet retrieval stage can be fed back to the search engine as an expanded query to harvest more relevant information; reinforcement learning can thus be utilized in this scenario.

For Phase B, we see that our system achieved five of six highest performance scores among all participating systems for factoid and list question answering in Batches 3, 4, and 5. We notice that
Table 6.5: Partial official BioASQ 3B evaluation result and ranks among systems (as of the manuscript completion) shown in the parentheses.

<table>
<thead>
<tr>
<th>Phase A: Document</th>
<th>Batch</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>MAP</th>
<th>GMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3rd</td>
<td>.2310 (15)</td>
<td>.3242 (15)</td>
<td>.2311 (15)</td>
<td>.1654 (15)</td>
<td>.0136 (15)</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>.2144 (15)</td>
<td>.3320 (15)</td>
<td>.2263 (15)</td>
<td>.1524 (15)</td>
<td>.0081 (14)</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>.2130 (15)</td>
<td>.4474 (15)</td>
<td>.2605 (15)</td>
<td>.1569 (15)</td>
<td>.0267 (8)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase A: Snippet</th>
<th>Batch</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>MAP</th>
<th>GMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3rd</td>
<td>.1133 (3)</td>
<td>.1044 (5)</td>
<td>.0891 (3)</td>
<td>.0892 (1)</td>
<td>.0013 (5)</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>.1418 (5)</td>
<td>.1264 (10)</td>
<td>.1153 (8)</td>
<td>.0957 (5)</td>
<td>.0027 (6)</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>.1472 (9)</td>
<td>.1756 (9)</td>
<td>.1391 (9)</td>
<td>.1027 (9)</td>
<td>.0040 (5)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase B: Exact Answers</th>
<th>Batch</th>
<th>Factoid</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strict Acc.</td>
<td>Lenient Acc.</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
<td>.1154 (1)</td>
<td>.2308 (1)</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>.4483 (1)</td>
<td>.6207 (1)</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>.2273 (1)</td>
<td>.3182 (1)</td>
</tr>
</tbody>
</table>

the performance in Batch 4 is higher than in other batches, which we believe is because Batch 4 set contains more questions seeking for the types of answers that have occurred more frequently in the training set, e.g. gene, disease, etc.

To further understand what causes the error and how we may improve the system, we manually answer each factoid question in Batches 3, 4, and 5 using the gold standard snippets provided for the input of Phase B, and compare with the output of our system to label the error types (multiple types allowed) for each incorrectly answered question. We list the error categories and give definition and examples to each category in Table 6.6, where we also show the occurrence of each error category in each test batch.

Table 6.6: Error categories and occurrences for factoid questions in BioASQ 3B test batches 3, 4, and 5.

<table>
<thead>
<tr>
<th>Error category</th>
<th>Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept type identification/answer type prediction</td>
<td>3rd 4th 5th</td>
</tr>
<tr>
<td>The highest ranked answer has a different concept type from the answer type that question asks for, which may be caused by a wrongly predicted answer type, an incorrect score combination equation from the score prediction model, or the concept identification module.</td>
<td></td>
</tr>
</tbody>
</table>

| Concept identification | 4 4 2 |
| Continued on next page | |
Table 6.6 – continued from previous page

<table>
<thead>
<tr>
<th>Error category</th>
<th>Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3rd</td>
</tr>
<tr>
<td>Some answer variants are not identified as concepts or we can find little evidence from the relevant snippets for the concept. For example, for the question Neurostimulation of which nucleus is used for treatment of dystonia? none of the components is able to identify Bilateral globus pallidus internus (GPi) as a concept and further candidate answer variant.</td>
<td></td>
</tr>
<tr>
<td><strong>Complex answer</strong></td>
<td></td>
</tr>
<tr>
<td>The ideal answer is a complex phrase or sentence, rather than a single-entity concept, usually in response to the questions containing effect, role, function, etc. For example, executors/mediators of apoptosis should be extracted to answer the question What is the function of caspases? but we only see “apoptosis” in the candidate answer list.</td>
<td>2</td>
</tr>
<tr>
<td><strong>Mistakenly use question phrase as answer</strong></td>
<td></td>
</tr>
<tr>
<td>Although we design a scorer in the ranking module to identify whether each candidate answer co-occurs in the original question, which should lower the rank of those candidate answers, we still see some question phrase variants are chosen as the top answer. For example, the question What is the effect of enamel matrix derivative on pulp regeneration? mentions a concept enamel matrix derivative but the system ranks its acronym “EMD” at the top.</td>
<td>3</td>
</tr>
<tr>
<td><strong>Tokenization</strong></td>
<td></td>
</tr>
<tr>
<td>Tokenization module may fail if the concept contains punctuation marks, e.g. parentheses, colon, semicolon, etc, and/or numbers, as in the example t(11;22)(q24;q12)</td>
<td>2</td>
</tr>
<tr>
<td><strong>Definition question</strong></td>
<td></td>
</tr>
<tr>
<td>The asker knows the terminology but asks for the definition, e.g. What is Piebaldism? or knows the properties and asks for terminology, e.g. How are ultraconserved elements called when they form clusters? We believe we need to introduce special question types and modules.</td>
<td>2</td>
</tr>
<tr>
<td><strong>Question type</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Identification of QUANTITY and CHOICE questions may fail in some cases. For example,
Alpha-spectrin and beta-spectrin subunits form parallel or antiparallel heterodimers?
does not use “Do” at the beginning. Another example is that “risk” is a QUANTITY indicator in
the question
What is the risk of developing acute myelogenous leukemia in Fanconi anemia?

Snippets that have no information
0 0 2
Some snippets do not contain any answer variant. For example,
What is the main role of Ctf4 in dna replication?
only has a gold standard snippet
Ctf4 remains a central player in DNA replication.

Relation concept identification
0 1 1
A relation concept refers to a verb or verbal adjective, e.g.
responsible or leading
that distinguishes the expected answer from other candidates that have the same concept type.

Syntactic function
0 1 1
The key to answer the question is embedded in the syntactic structures of the relevant snippets.
For example, in the snippet
Medicarpin, the major phytoalexin in alfalfa, is synthesized via the isoflavonoid branch of phenylpropanoid metabolism.
no explicit relation word is used between “Medicarpin” and “the major phytoalexin”, but the
syntactic structure clearly implies that the latter explains the former.

Based on the analysis, we believe a better concept identification model and concept type pred-
diction model will make the hugest impact to the overall performance improvement. Moreover,
we plan to conduct a thorough ablation study to estimate how much each component or feature
contributes to the overall performance, as soon as we have the gold-standard outputs for the 3B
dataset.

6.6 Analytics Space Exploration

We plan to integrate all the analysis components and perform process optimization using the prop-
osed exploration strategies.

[PROPOSED WORK, C.F. SECTION 6.7 AND CHAPTER 9]
6.7 Proposed Work

1. We continue to create analysis components for the achieved functionalities to improve their performance and develop components for the missing functionalities, such as yes/no QA, summary generation, etc., to complement the current system.

2. We then proceed to leverage the proposed exploration strategies to boost the system performance using the BioASQ 2013–2015 benchmark set containing over 1K decision scenarios nominated and their gold standard solutions created by biomedical professionals [154].
Chapter 7

Case Study: Pharmaceutical Decision Task

Developing a new drug is a complex and costly process that proceeds from the identification of a potential therapeutic candidate to marketing a drug product, usually taking more than a decade and costing in excess of 1 billion US dollars [65, 127]. This includes identify and properly validate drug targets, define the utility of employing probes in the early discovery phase, medicinal chemistry, lead optimization, preclinical proof of concept strategies, and drug delivery needs through preclinical proof of concept [66]. There exist various approaches in drug development: New Chemical Entity (NCE) discovery, drug repurposing (beneficial activity currently marketed drugs possess against novel targets), drug delivery improvement, etc. [66].

Along each development path, decision makers need to consider a series of predetermined criteria at each phase, in order to prioritize drug candidates to reduce risk to human subjects and to increase the chance of picking a winning therapeutic molecule in NCE [127]. For example, in the earliest stage of NCE, target identification and target validation (or target assessment) aim to select and prioritize a number of disease targets (agents with a particular biological action that are anticipated to have therapeutic utility), and estimate the “druggability” of each target influenced by a complex balance of scientific, medical and strategic considerations, including efficacy, safety, commercial profits, etc. [65, 77]. A huge amount of structured or unstructured data have become available and can facilitate the stakeholders to make better prediction and decision. For example, one can review the literature and public/proprietary data to collect a list of molecule modulating targets, know the history of success of each type of target, and obtain genetic confirmation. He/she can also investigate the intellectual property and marketability of the target from patent database and market analysis portals [66].

Discovering and evaluating all the relevant, but possibly inconsistent or even contradictory, conclusions from over 20 million biomedical research publications as well as various manually curated databases can be very challenging. We attempt to leverage analytics meta learning to help with such complex decision support tasks. In this case study, we focus on the scientific factors rather than the commercial aspects of the problem. We first review the specialized decision support systems in Section 7.1. In Section 7.2, we formally define the use case of the problem by showing inputs and expected outputs. In Section 7.3, we first present a manually created analytics procedure for target validation, which includes a decision process and a simple generic execution process for each decision factor. Then, we try to automatically construct the procedure space for a
series of pharmaceutical decision tasks. In Section 7.4, we describe a solution synthesis algorithm and the analysis components that we have developed for analysis steps in the execution process for the target validation support task, which we plan to replace with the general biomedical question answering system developed in Chapter 6. In Section 7.5, we propose to apply a full analytics meta learning approach to construct and optimize a general-purpose pharmaceutical decision support system. In Section 7.6, we propose to take advantage of biomedical documents to extract human decision processes for various decision scenarios, and then outline the necessary steps to conduct a comprehensive exploration of all possible procedures and configurations across components.

### 7.1 Related Work

Many decision support systems have been developed in support of various decision scenarios during the past few decades [21, 92, 167]. Although most decision support systems are built for predetermined tasks, which requires domain knowledge obtained from humans and stored in a structured data store with a predefined schema, a few decision support systems have explored how to understand decision needs from natural language input. For example, the Structured Evidential Argumentation System (SEAS) [92] provides decision references for national security crisis warnings, and Intelligent Decision System (IDS) [167] analyzes business innovation self-assessments written in free-text. However, neither approach is general or extensible enough to cover the cases we implemented with our proposed solution and open-source implementation.

### 7.2 Task Description

Similar to biomedical information seeking needs described in Section 6.2, pharmaceutical decision tasks have the same input type, e.g. a factoid/list question

```
Which genes are directly involved in the breast cancer and can be a suitable target?
```

or a yes/no question

```
Is FGFR1 directly involved in the breast cancer and can be a suitable target?
```

and output requirement

```
AKT1, BRCA1, BRCA2, BRIP1, ERBB2, FGFR1, KDR1, mTOR, NBN, FGFR1, KDR1, mTOR, etc.
```

or

```
yes
```

102
Since the users are often interested in understanding how the decision is made, a detailed decision report is also needed, which includes the decision logic and the collected evidence, either formatted or in natural language, e.g.

- If there exists any in vivo experiment showing that modulating the activity of the gene affects biochemical function or phenotype of breast cancer, then it is a positive sign that the gene can be a target.
- We have found the literature that there exists in vivo experiment showing that modulating the activity of FGFR affects biochemical function or phenotype of breast cancer, including
  - PMID: 25400686
  - PMID: 15863030
  - ... 
  - ...
  - ...

7.2.1 Evaluation metrics

Although the outcome in the decision support tasks is sometimes subjective and difficult to measure until later phases are conducted in the drug development process, this case study, as the first attempt to a series of pharmaceutical decision scenarios, demonstrates how to make prediction purely and automatically from the literature by considering various decision factors involved in the decision. We use the same set of evaluation metrics as for the list questions [32] in Chapter 6 which include precision, recall, and F-1 when the order is ignored, as well as mean average precision (MAP) for retrieval tasks when the order is considered. Support target validation, We take target validation support as an example. The list of the genes produced by the system and ranked by the confidence score can be compared against the unordered known target list for the disease.

7.2.2 Dataset

Collecting benchmark dataset (tasks and golden results) for pharmaceutical decision scenarios is difficult. We mainly focus on two types of sources. First, we select a subset of analytics tasks related to pharmaceutical decisions from the BioASQ question set [154], and reuse the provided golden answers, whereas we take the full advantage of analytics meta learning to create a solution based on an optimal analytics procedure that not only considers execution processes (as in Chapter 6) but also decision processes.

Second, we collect research questions manually and automatically from related research articles including tutorials, and then we either manually judgment the credibility of each answer or turn to manually curated databases. For example, to judgment the “drugability” for target validation,
we used the DisGeNET 2.0 dataset [11], which contains 6,029 diseases and 9,313 genes integrated from a number of gene-disease association databases, including OMIM\(^1\) UniProt\(^2\) PharmGKB\(^3\) and CTD\(^4\) The DisGeNET dataset was used for QUADS model training and evaluation, and we therefore focus on the manually-created items and exclude the LHGDN subset, which contains gene-disease relations automatically extracted by text mining algorithms. We split the manually created dataset into 5,605 subsets, each of which corresponds to one of the remaining diseases, and 6,158 genes were used as candidates for target validation.

7.3 Analytics Procedure Definition

In the first half of this section, we introduce a manually created decision process for target validation, which is then expanded into a full analytics procedure. In the second half, we leverage the procedural knowledge discovery algorithms proposed in Chapter 3 to automatically propose the analytics procedure candidates for pharmaceutical decision tasks, which enables analytics meta learning to optimize over a larger analytics space.

7.3.1 Manual Analytics Procedure for Target Validation

We created a simple target validation decision process to test our hypothesis, based on literature review as well as personal knowledge and experience from a group of professional biochemists and bioinformaticians working at a pharmaceutical company. We first defined GENE and DISEASE as the variables in the target validation DPT\(^5\) which could then be instantiated with actual GENE and DISEASE names from the relations in the DisGeNET dataset. We expanded the template containing questions in the form of Jeopardy! clues in a top-down fashion, which, depending on the use case, can be treated as a yes/no question if both GENE and DISEASE are instantiated and a list question if only DISEASE is instantiated. Specifically, we started with defining the decision goal, which summarizes the target identification task from a high level:

The GENE is directly involved in DISEASE and can be a suitable target.

As one can easily see that necessary detailed criteria for a “druggable” target cannot be thoroughly depicted in the decision goal, the template was then expanded to a second level, which contains six important decision factors such as gene expression, gene mutation, pathway, clinical trials, etc., that was further decomposed as necessary. Finally, the decision process contains a total of 16 factors in three levels, which is shown in Figure 7.1. We can see from Figure 7.1 that some strong indicators can hardly be satisfied by a single gene, whereas some other weak factors only

\(^1\)Online Mendelian Inheritance in Man Database, [http://omim.org](http://omim.org)
\(^2\)Universal Protein Resource Database, [http://www.uniprot.org](http://www.uniprot.org)
\(^3\)Pharmacogenomics Knowledge Base, [http://www.pharmgkb.org](http://www.pharmgkb.org)
\(^4\)Comparative Toxigenomics Database, [http://ctdbase.org](http://ctdbase.org)
\(^5\)Conventionally, a gene can refer to either a gene or a gene product such as protein or mRNA.
1 The GENE is directly involved in the DISEASE and can be a suitable target.

1.1 Any experiment showing that modulating the activity of the GENE with a chemical compound or genetic modification causes the DISEASE.

1.1.1 Any human in vivo experiment showing that modulating the activity of the GENE affects biochemical function or phenotype of the DISEASE.

1.1.2 Any in vitro experiment showing that modulating the activity of the GENE affects biochemical function or phenotype of the DISEASE.

1.1.3 Any animal model study showing that modulating the activity of the GENE in animals causes the DISEASE.

1.2 The GENE is expressed in the human tissue related to the DISEASE.

1.2.1 The GENE is expressed in normal human tissue related to the DISEASE.

1.2.2 The GENE expression is altered in human DISEASE tissue or human DISEASE cell.

1.2.3 The alteration of the GENE expression is correlated with the DISEASE severity.

1.3 Any mutation is associated with the DISEASE.

1.3.1 Any mutation of the GENE has significantly associated with the DISEASE.

1.3.2 Any mutations in other genes linked to the GENE associated with DISEASE.

1.4 Any pathway involving the GENE supports that the GENE causes the DISEASE.

1.5 Any clinical trials show that targeting the GENE can prevent or slow the progress of the DISEASE.

1.6 Any evidence suggests that targeting the GENE will have side effects.

1.6.1 Targeting the GENE will cause liver, heart, and kidney damage.

Figure 7.1: Decision subtasks in the three-level target validation task
contribute to the decision if some gene satisfies all the criteria. We also pay special attention to the criteria 1.6 and 1.6.1, which intuitively are negative indicators. We did not specify any preferences to the decision processes as input, i.e. a uniform prior was used by default, and we count on the decision logic to decide what weights should be assigned when combining the factors.

We further expand the decision process to a full analytics procedure by decomposing each decision factor with an execution process similar to a supporting evidence based QA system such, which commonly incorporates parsing, interpretation, retrieval, answer extraction, evidencing and ranking phases. Although we design to use the same general execution procedure to deal with all the sub analytics tasks at each decision factor, and we further utilize the same set of components to instantiate the execution process, the resulting analytics engines are still different and specific to each factor, due to the difference of configurations. We will detail this in Section 7.4.

7.3.2 Automatic Analytics Procedure Construction

We perform the proposed automatic procedure construction algorithms to the selected pharmaceutical decision tasks according to Section 3.5 and compare the output with human decision processes.

7.4 Analysis Component Construction

This section focuses on instantiating the analytics procedure with analysis components. We first describe the analysis components in the yes/no question answering pipeline. Then, we describe the modifications to the general biomedical question answering system when adapting it to solve the sub analytics tasks for the pharmaceutical decision tasks.

7.4.1 Supporting Evidence based Yes/No QA Analysis Components

We deal with the analytics tasks represented as yes/no questions, i.e. either \( y \) or \( n \) will be assigned to label each assertion at the end of the pipeline. We list the key components of the QA pipeline in Table 7.1. After a question analysis phase which introduces NLP annotations, different queries are constructed from all or part of the extracted focus terms, and other nouns, verbs, and named entities (with synonyms) are also combined via an OR operator. A Lucene index was used to retrieve candidate passages based on various queries, and document frequencies for these queries were then combined with weights learned using logistic regression.

Compared with arbitrary nominal assertions, yes/no assertions can generalize the learned decision model\(^6\). We can see that the chosen QA pipeline lacks the algorithmic sophistication (i.e. inference capability) required to automatically interpret and match the term “suitable target” by filling in necessary information related to the specific factors.

\(^6\)One can also consider different levels of confidence as assertions, e.g. certain, probable, probably not, impossible, etc.
### Table 7.1: Yes/no QA analysis components

<table>
<thead>
<tr>
<th>Category</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsing</td>
<td>LingPipe HMM based tokenizer</td>
</tr>
<tr>
<td></td>
<td>LingPipe HMM based POS tagger</td>
</tr>
<tr>
<td></td>
<td>LingPipe HMM based named entity recognizer</td>
</tr>
<tr>
<td></td>
<td>Rule based lexical variant generator</td>
</tr>
<tr>
<td>Interpretation</td>
<td>UMLS for disease syn/acronym expansion</td>
</tr>
<tr>
<td></td>
<td>EntrezGene for gene syn/acronym expansion</td>
</tr>
<tr>
<td>Retrieval</td>
<td>Lucene search engine</td>
</tr>
<tr>
<td>Evidencing and ranking</td>
<td>Document frequency (DF)</td>
</tr>
<tr>
<td></td>
<td>Weight learning with logistic regression</td>
</tr>
</tbody>
</table>

#### 7.4.2 Adapting General Biomedical QA

In this section, we employ the general-purpose biomedical components and pipelines developed in Chapter 6 to perform sub analytics tasks, which replaces the original target validation QA pipeline, with the intention of covering a wider range of pharmaceutical decision scenarios.

[PROPOSED WORK, C.F. SECTION 7.6 AND CHAPTER 9]

#### 7.5 Analytics Space Exploration

In this section, we explore the configuration space that contain analytics procedures extracted in Section 7.3, where we also investigate how analytics meta learning advances rapid domain adaptation, i.e. given a system pre-optimized for an existing analytics task, how we can quickly adapt the system to pharmaceutical decision tasks.

[PROPOSED WORK, C.F. SECTION 7.6 AND CHAPTER 9]

#### 7.6 Proposed Work

1. First, we perform the proposed automatic procedure construction algorithms to selected pharmaceutical decision tasks (e.g. target validation), and compare the output with human decision processes (e.g. target validation decision process).

2. Second, we employ the general-purpose biomedical components and pipelines to perform sub analytics tasks, which replaces the original target validation QA pipeline, with the intention of covering a wider range of pharmaceutical decision scenarios.

3. Third, we explore the analytics space that contain massive analytics procedures, where we also investigate how analytics meta learning advances rapid domain adaptation, i.e. given a system pre-optimized for an existing analytics task (e.g. BioASQ task), how we can quickly adapt the system to a related but different analytics task (e.g. drug discovery).
Chapter 8
Case Study: Product Recommendation Task

This chapter continues the study of analytics meta learning applications, and focuses on another use case that involves complex problem solving procedure – product recommendation task. Unlike the target validation problem, algorithmic methods for the product recommendation problem (as well as the related rating problem) have been extensively investigated [97]. In this section, rather than propose any new algorithm for this relatively well-studied problem, we focus on utilizing analytics meta learning methodology to automatically construct an information system, in order to test its flexibility and generalizability.

In Section 8.1, we give formal definition to the product recommendation task, and in Section 8.2, we describe the manual decision process and propose to take advantage of purchase guide documents to extract human decision processes for various decision scenarios. In Section 8.3, we describe the analysis components that we have developed for the phone recommendation task. We report the initial results using the proposed framework that incorporates a manually created decision process in Section ???. In Section 8.4, we propose to apply a full analytics meta learning approach to construct and optimize a general-purpose product recommendation system using review texts and product databases. In Section 8.5, we outline the steps to conduct a comprehensive exploration of all possible procedures and configurations across components.

8.1 Task Description

Similar to the pharmaceutical decision task introduced in Chapter 7 and formally defined in Section 7.2, the product recommendation task is an analytics task that real users have to deal with every day online and offline. For example, one can tell the system the decision needs in natural language, e.g.

Is buying an iPhone 5 recommended?

or

Which PHONE is recommended to purchase?
and the answer can be

yes

or

iPhone 6s, Samsung Galaxy s6, etc.

Users also expect to find out the supporting evidence from the review texts (e.g. user experience, quantity, etc.) and/or specification database (e.g. weight, color, operating system, etc.).

8.1.1 Evaluation metrics

Since this task is more subjective than target validation, we can hardly rely on a manually-created gold-standard product recommendation list. Thus we use the average rating of each product on Amazon as its overall recommendation weight, which is then converted from a 5-point rating scale to a binary recommendation. Specifically, ratings equal to or smaller than 3 are assigned no, otherwise yes. In this way, we can collect a set of golden “recommended” products for each category, which are compared with a ranked list of products recommended by the system. Thus, we can utilize the same set of evaluation metrics including (mean) precision, (mean) recall, and (mean) F-1, and AP (MAP) as in Chapter 7.

8.1.2 Dataset

We employed the Amazon product review corpus. Various product reviews give insights about the same product from different perspectives, in the same way that different publications in PubMED discuss different perspectives of the same gene/disease. Similar to the preparation procedure for PubMED abstract corpus, we here focused on the review/text field in each review and tagged with productId. We might also consider other fields such as userId or time, which provide supplementary information for the recommendation task. However, incorporating structured data with a different schema often requires extra effort to adapt existing QA pipelines.

8.2 Analytics Procedure Definition

In this section, we first describe the manually created decision process, and then propose to take advantage of purchase guide documents to extract human decision processes for various decision scenarios.

8.2.1 Manual Decision Process for Buying a Phone

We first present a manual decision process for buying a phone. Unlike the target validation task, where a few second-level aspects can cover most considerations for a majority of diseases, different
products have different decision factors, which motivated us to focus on a particular product type (cell phones). We allowed QA pipelines to return nominal answer texts (e.g. Apple, Samsung, Nokia, Google, etc. from a finite set of brands). Compared with the number of cell phone products and reviews available in the data set, this restriction did not cause any issues with data sparsity.

When creating a decision for cell phone recommendation, we first defined the decision goal as

Buying the PHONE is recommended.

where PHONE is the variable in the template. Then, we considered the most important high-level features, including

- design and usability
- brand
- functionality
- carrier
- operating system
- ...

and we created natural language questions corresponding to each feature, e.g.

The PHONE has good usability.

or

The PHONE is made by Apple.

For broad questions such as the one asking about usability, we further decomposed the question down to two sub-factors:

The PHONE is light.

The PHONE has a good interface design.

Finally, we created a DPT with 17 decision factors.

### 8.2.2 Automatic Analytics Procedure Construction

Similar to the pharmaceutical decision task, we perform the proposed automatic procedure construction algorithms introduced in Section 3.6 to selected product recommendation tasks, and compare the output with human decision processes.

[Proposed Work, c.f. Section 8.5 and Chapter 9]
8.3 Analysis Component Construction

This section focuses on describing the components.

8.3.1 Adapting simple yes/no question answering pipeline

We applied almost the exact same set of QA components for this task as for Chapter 7. However, some changes were required. First, we replaced the POS tagging model and NER model to models trained on standard English news corpora. Second, we replaced the biomedical synonym expansion component with a sentiment word dictionary. We also integrated the index for the reviews to replace the biomedical corpus. The rest of the pipeline remains the same.

8.3.2 Adapting general question answering components

In this section, we adapt the analysis components developed in the BaseQA layer (c.f. Section 6.3), which is designed to contain the general-purpose analysis components and configurations, to the product recommendation task.

8.4 Analytics Space Exploration

In this section, we explore the configuration space that contain analytics procedures, where we also compare the results with the analytics meta learning applications described in Chapters 6 and 7.

8.5 Proposed Work

1. First, similar to the pharmaceutical decision task, we perform the proposed automatic procedure construction algorithms to selected product recommendation tasks, and compare the output with human decision processes.

2. Second, we collect product recommendation related data sources and develop analysis components to each specific type of data, as part of a general-purpose QA framework – BaseQA.

3. Third, we explore the configuration space that contain massive analytics procedures, where we also compare the results with the other analytics meta learning applications.

3 http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar
Chapter 9

Conclusion

This thesis proposal has set up an ambitious goal of unifying the analytics meta learning tasks in a single framework. Based on the amount of the work that has been accomplished and published, this is a reasonable work. Here, we summarize the current progress towards the completion of the thesis in each respect chapter and present the schedule for accomplishing the remaining work.

Table 9.1: Tasks in the theoretical part with the status: ✔ for completed task, ✘ for work in progress, ☐ for pending and requires investigation.

<table>
<thead>
<tr>
<th>Mission</th>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics meta learning problem</td>
<td>Terminology and problem definition</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Solution framework and problem breakdown</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Representation language for configuration space (ECD)</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Representation language for decision processes (DPT)</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Unified procedure representation</td>
<td>✔</td>
</tr>
<tr>
<td>Procedural knowledge discovery</td>
<td>Procedural knowledge (base) definition</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Procedural knowledge extraction from semi-structured procedural knowledge base</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Applying semi-structured procedural knowledge base for query suggestion</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Procedural knowledge discovery from search log and suggested queries</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Procedural knowledge discovery from community QA sites</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Procedural knowledge discovery from unstructured texts (e.g. bibliographic DBs)</td>
<td>✔</td>
</tr>
<tr>
<td>Exploration algorithms</td>
<td>Configuration space exploration strategy for pipelines</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Preliminary experiment on biomedical passage retrieval system</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>General strategies for analytics space exploration</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Theoretic analysis of asymptotic and dynamic properties</td>
<td>✘</td>
</tr>
</tbody>
</table>
Table 9.2: Tasks in the empirical part with the status: ✔️ for completed task, ⚫️ for work in progress, ⚫️️ for pending and requires investigation.

<table>
<thead>
<tr>
<th>Mission</th>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics meta</td>
<td>Design and implementation principles</td>
<td>✔️</td>
</tr>
<tr>
<td>learning implementation</td>
<td>Supporting unified procedure representation</td>
<td>⚫️</td>
</tr>
<tr>
<td></td>
<td>Integrating exploration strategies</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Integrating procedural knowledge discovery algorithms</td>
<td>⚫️️</td>
</tr>
<tr>
<td>Biomedical question answering task</td>
<td>Task definition</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Analytics procedure definition</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Exact answer generation components for factoid and list tasks</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Exact answer generation components for yes/no tasks</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Ideal answer generation components</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Documents as supporting evidence gathering components</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Passages as supporting evidence gathering components</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Concepts as supporting evidence gathering components</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Performing configuration space exploration</td>
<td>⚫️️</td>
</tr>
<tr>
<td>Pharmaceutical decision task</td>
<td>Task definition</td>
<td>⚫️</td>
</tr>
<tr>
<td></td>
<td>Performing automatic procedure construction</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Integrating biomedical analytics task solutions</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Performing analytics space exploration</td>
<td>⚫️️</td>
</tr>
<tr>
<td>Product recommendation task</td>
<td>Task definition</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Performing automatic procedure construction</td>
<td>⚫️</td>
</tr>
<tr>
<td></td>
<td>Developing general analysis components</td>
<td>⚫️️</td>
</tr>
<tr>
<td></td>
<td>Performing analytics space exploration</td>
<td>⚫️️</td>
</tr>
</tbody>
</table>
Figure 9.1: Gantt chart for the proposed work
Appendix A

YAML Descriptors

A.1 Biomedical Question Answering Case Study

Listing A.1: ECD main descriptor for test batch 5 in Phase B

```yaml
# execute
# mvn exec:exec -Dconfig=bioasq.test
# to test the pipeline

configuration:
  name: test
  author: ziy

persistence-provider:
  inherit: baseqa.persistence.local-sqlite-persistence-provider

collection-reader:
  inherit: baseqa.collection.json.json-collection-reader
dataset: BIOASQ-QA
file:
  - /input/3b-5-b.json
type: [factoid, list, yesno, summary]
decorators: |
  - inherit: bioasq.gs.bioasq-qa-gs-decorator
  persistence-provider: |
    inherit: baseqa.persistence.local-sqlite-persistence-provider

pipeline:
  - inherit: ecd.phase
    options: |
      - inherit: bioga.quesanal.parse-clearnlp-bioinformatics
    - inherit: ecd.phase
    options: |
```
- `inherit`: bioqa.quesanal.concept-metamap
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.quesanal.concept-lingpipe-genia
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: baseqa.quesanal.concept-opennlp-np
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: baseqa.quesanal.concept-opennlp-nppnnp
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.quesanal.lexical-answer-type
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.quesanal.at.predict-liblinear
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: baseqa.retrieval.passage-to-view
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.retrieval.passage-parse-clearnlp-bioinformatics
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.retrieval.passage-concept-metamap
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.retrieval.passage-concept-lingpipe-genia
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: baseqa.retrieval.passage-concept-opennlp-np
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: baseqa.retrieval.passage-concept-opennlp-nppnp
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.retrieval.concept-search-uts
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: baseqa.retrieval.concept-merge
- `inherit`: ecd.phase
  `options`: |
  - `inherit`: bioqa.answer.generate
- `inherit`: ecd.phase
  `options`: |
post-process:
  
  # answer evaluation
  - inherit: baseqa.eval.base
calculator: |
    - inherit: bioasq.eval.calculator.answer-eval-calculator
evaluatee-provider: |
    - inherit: baseqa.eval.evaluatee.answer-evaluatee-provider
  persistence-provider: |
    - inherit: baseqa.eval.persistence.jdbc-eval-persistence-provider

  # report
  - inherit: report.csv-report-generator
  builders: |
    - inherit: baseqa.report.accumulated-measurements-report-component

  # submission
  - inherit: bioasq.collection.json.json-cas-consumer

Listing A.2: ECD component descriptor of bioqa.answer.generate

class: edu.cmu.lti.oaqa.baseqa.answer.CavGenerationManager

generators: |
  - inherit: baseqa.answer.generators.choice
  - inherit: baseqa.answer.generators.quantity
  - inherit: bioqa.answer.generators.concept
  - inherit: baseqa.answer.generators.cav-covering-concept
  - inherit: baseqa.answer.generators.yesno

Listing A.3: ECD component descriptor of bioqa.answer.score-predict-liblinear

inhibit: baseqa.answer.score-predict

classifier: ‘inhibit: bioqa.answer.score-classifier-liblinear’
scorers: |
  - inherit: baseqa.answer.scorers.type-coercion
  - inherit: baseqa.answer.scorers.cao-count
  - inherit: baseqa.answer.scorers.name-count
  - inherit: baseqa.answer.scorers.avg-covered-token-count
  - inherit: baseqa.answer.scorers.stopword-count
- `inherit`: baseqa.answer.scorers.token-overlap-count
- `inherit`: baseqa.answer.scorers.concept-overlap-count
- `inherit`: baseqa.answer.scorers.token-proximity
- `inherit`: baseqa.answer.scorers.concept-proximity
- `inherit`: baseqa.answer.scorers.lat-overlap-count
- `inherit`: baseqa.answer.scorers.parse-proximity
Bibliography


[84] Yanen Li, Bo-June Paul Hsu, and ChengXiang Zhai. Unsupervised identification of syn-


1982.


[152] the Apache UIMA Development Community. Uima tutorial and developers’ guides, August 2015.


Di Xu and Jamie Callan. Towards a simple and efficient web search framework, 2014.


