INTELLIGENT AGENT DEVELOPMENT USING UNSTRUCTURED TEXT CORPORA AND MULTIPLE CHOICE QUESTIONS

By

Joseph Johnson

A Dissertation Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: COMPUTER SCIENCE

Examining Committee:

_____

Selmer Bringsjord, Dissertation Adviser

_____

Micah Clark, Member

_____

Eugene Eberbach, Member

_____

Sergei Niremberg, Member

_____

Mei Sei, Member

_____

Boleslaw Szymanski, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2016
(For Graduation August 2016)

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

I'd like to extend my sincere gratitude to my advisor, Prof. Selmer Bringsjord, for his continual support and encouragement during my doctoral studies – Thank you, Selmer! I'd also like to thank the members of my dissertation committee – Dr. Micah Clark, Prof. Eugene Eberbach, Prof. Sergei Nirenburg, Prof. Mei Sei, and Prof. Boleslaw Szymanski – for their guidance over the course of completing my dissertation. I'd also like to thank the staff of the RPI Computer Science and Cognitive Science departments – Paula Monihan, Terry Hayden, and Sibel Adali – for their support in navigating the twists and turns of the administrative process of the Ph.D. program. I'd also like to give a special thanks to Naveen Sundar Govindarajulu, who as a colleague during my graduate work, provided sage advice, insight, and wisdom during this long journey.

Most importantly, I'd like to extend my greatest thanks to my wife, Courtney, and my daughters, Kayla and Avery, without whose unquestioning support and patience this endeavor would not have been possible.

ABSTRACT

This dissertation explores various approaches for developing an intelligent agent in a particular domain: fraud detection. The framework by which we measure our agent is *psychometric artificial intelligence*, or, more commonly, *psychometric AI* which focuses on the creation of agents that can successfully pass tests of mental ability. This approach offers a number of benefits, including a well-defined domain, a built-in measure for quantifying the efficacy of the agent in the form of a test score, and a rich environment for deploying various forms of AI, including machine learning, natural language processing, computer vision, et cetera. In this work, our attention we'll be centered around natural language processing (NLP).

As part of our commitment to this psychometric approach, we set our sights on one particular test in the fraud-detection domain – the Certified Fraud Examiners (CFE) exam, administered by the Association of Fraud Examiners (ACFE), the governing body overseeing the fraud-examiners profession. The CFE exam is a multiple-choice exam whose questions are based on the material of the Fraud Examiners Manual (FEM), herein referred to as the CFE Manual. Programmatic processing of both the CFE Manual and of a training set of CFE exam questions generate the basis of the agent's knowledge and decisions for answering new questions on the test.

The approaches employed in the work presented herein range over a variety of techniques, from shallow text-processing techniques to deep, cognitive, semantic-representation techniques. Version 1 of the agent focuses on shallow text-processing techniques that leverage features of the exam and the high-level structure of the CFE Manual. Analysis of these shallow algorithms on a training set is then used to apply these algorithms optimally on a test set. As we'll discuss, even these relatively simplistic techniques generate surprisingly decent scores, although not ones at the level of passing. Version 2 employs more sophisticated techniques than Version 1, using information-retrieval approaches to the question-answer task, wherein the agent decomposes the CFE Manual into a hierarchical structure of granular

documents using the CFE Manual's table of contents and text features. Version 3 incorporates machine learning to target the precise paragraphs within the CFE Manual relevant to each question. Finally, Version 4 features an agent with a deep, semantic representation of a subdomain of fraud detection, doctor shopping, and whose knowledge-base consists of assertions expressed in the *deontic cognitive event calculus* $\mathcal{DCEC}^*$. The intent in this version is to demonstrate for an example subdomain of fraud detection how formal representation and logico-deductive methods provide transparent justifications and fraud detection.

This dissertation attempts to advance the field of AI in three ways: First, introduce and measure the performance of new algorithms in the AI subfield of question-answer (QA). Second, provide a comparative analysis of these approaches in terms of performance and their ability to provide reasoning justification, and by doing so address an important question in QA – What are the various marginal costs/benefits associated with state of the art AI approaches in terms of the competing priorities of accuracy, complexity, and provision for reasoning justification. And third, lay the groundwork for the rigorous study of fraud detection using the formal representation and reasoning approach demonstrated in Version 4, which is not an agent to be measured against exam questions, but one that provides a demonstration of the formal approach to fraud. It is hoped that by the end of this exploration, the reader will have a solid understanding of the various techniques discussed herein, an appreciation of the benefits of using psychometric AI as the backdrop for intelligent-agent development, and some insights into the relative potential and costs of each of these approaches.

CHAPTER 1
Introduction and Overview

1.1 Background

Question Answer (QA) [2, 1] is a prominent and growing sub-field of natural language processing, and of the larger field of AI [3]. QA systems [1] consist of agents that provide responses (correct ones, hopefully) to questions posed in natural language. These systems are typically based on one of two approaches: an information-retrieval approach [4, 5, 6, 7] or a knowledge-based approach [8, 9, 10]. Some of the most successful systems as of late, however, utilize a blend of these two approaches, such as IBM's Watson [11]. A knowledge-based approach offers the advantages of allowing automated reasoning, and thus, justifications for answers. But the knowledge bases involved have been manually curated by domain experts and knowledge engineers – a time-consuming, labor intensive endeavor. An information-retrieval approach, where candidate answers are screened from passages retrieved from documents retrieved from a core information-retrieval (IR) system, are designed without the need to manually curate a knowledge base, but typically offer no justification of answers. In fact, these systems have no understanding of the reasons for the answers they put forth, but instead typically provide answers based on statistical approaches that are utilized in black-box fashion, giving the user no insight into the rationale for responses.

1.2 Goals

This dissertation explores various approaches for developing an intelligent agent in a particular domain, fraud detection, in the context of an artificial-intelligence framework, called *psychometric artificial intelligence*, or, more commonly, *psychometric AI. Psychometric AI* focuses on the creation of agents that can successfully

Portions of this chapter previously appeared as: J. Johnson, "Toward an intelligent agent for fraud detection – the CFE Agent," in *Deceptive and Counter-Deceptive Machines: Papers from the AAAI Fall Symposium*, no. FS-15-03. AAAI Press, 2015, pp. 21–26

pass tests of mental ability, and offers a number of benefits, including a well-defined domain, a built-in measure for quantifying the efficacy of the agent in the form of a test score, and a rich environment for deploying various forms of AI, including machine learning, natural language processing, computer vision, et cetera.

This dissertation attempts to advance the field of AI in three ways: First, introduce and measure the performance of new algorithms in the AI subfield of question-answer (QA) that provide justifications for their reasoning. Second, provide a comparative analysis of these approaches in terms of their accuracy, relative complexity, and their ability to provide justification for their reasoning by assessing three versions of the CFE Agent – Versions 1, 2, and 3 – in which these approaches are step-wise implemented. By doing so we address an important question in QA – What are the various marginal costs/benefits associated with state-of-the-art AI approaches in terms of the competing priorities listed above. And third, lay the groundwork for the rigorous study of fraud detection using the formal representation and reasoning approach demonstrated in Version 4.

1.3 The Intelligent Agent Defined, For This Project

In an effort to further detail the goals listed above, the definition of intelligent agent is made more explicit, here, as one capable not only of making intelligent decisions, but also of providing a justification for those decisions. The ideal for a justification is one that is proof-based as exemplified in [12] and [13]. Just shy of this ideal is a different approach that shows the premises for each of its conclusions, as in justification-based truth maintenance systems, (JTMS), as described in [14]. Finally, absent these, we'd like at least some explanation from the agent for its reasoning (as we'll discuss for Versions 2 and 3). This is to be contrasted with agents of other AI systems where decisions are provided typically without any justifications.[1]

[1]For example, in a facial recognition system, the agent typically matches identities to faces in photographs without any explicit explanation for its reasoning.

1.4 An Overview of the Approach

The overarching idea is to develop successively more sophisticated agents – starting with a naive agent that uses only shallow techniques that largely leverage question features with patterns discovered from exploring a training set. In Version 2, we elevate the level of sophistication of the agent, but still limit it to fairly shallow techniques, by basing answers on information-retrieval-based techniques that incorporate sophisticated query-generation and document-collection-development techniques. In Version 3, we explore machine learning approaches for a passage selection algorithm. And finally, Version 4 consists of a demonstration of the behavior of an agent, which, if it were implemented, would use deep, formal, semantic techniques to answer questions about a subdomain of fraud detection – doctor shopping.

1.4.1 Reasoning with Uncertainty

In order to cover the nondeterminism inherent when extracting information using NLP techniques, the agent incorporates uncertainty into its decisions. As we'll see in Version 1 and Version 2, the agent quantifies uncertainty by analysis of its various algorithms on the training set, and then utilizes this information to optimize its accuracy on the test set. It also quantifies uncertainty in some intermediate steps of its algorithms, such as in the paragraph selection algorithm in Version 3.

1.4.2 Verification of the Agent via Test-Taking

Algorithm performance is assessed by measuring accuracy on multiple-choice questions. In this project, the domain of fraud detection is used, for which there is a well-developed, effective industry for testing subjects on knowledge in this domain. The principal organization responsible for this and its examination process are described below.

1.4.2.1 The ACFE and the CFE Exam

The ACFE [15] describes itself on its website as "the world's largest anti-fraud organization and premier provider of anti-fraud training and education" [15]. Generally speaking, in order to become a readily employable expert in the field of fraud detection, certification by this organization is required. (Among notable certified

members of the ACFE is Harry Markopolos, the American forensic accountant who achieved fame by being the first to have uncovered the ponzi scheme perpetrated by Bernard Madoff and desperately tried to warn government securities officials years before the scam collapsed in the midst of the 2008 financial crisis [16].) The ACFE has well-defined requirements for becoming certified, based on a point system that considers a combination of professional experience and academic credentials. However, the CFE exam is the credentialing centerpiece for the ACFE, and the details of this exam are described briefly below.

The CFE Exam is a computer-based exam. The mechanics for preparing for and taking the exam begins with downloading a software package from the prep course page of the ACFE website. This package includes the exam software, the Fraud Examiners Manual (on which the test is based), and a self-study application consisting of a battery of sample test questions, a complete practice exam, and tools for monitoring progress.

The CFE exam consists of 4 sections, listed below:

1. Financial Transactions and Fraud Schemes
2. Law
3. Investigation
4. Fraud Prevention and Deterrence.

Each section consists of 125 multiple-choice and true-false questions. The candidate is limited to 75 seconds to complete each question and a maximum total allocated time of 2.6 hours to complete each section. Each CFE Exam section is taken separately. The timing for each section is subject to the candidates discretion. However, all four sections of the exam must be completed and submitted to the ACFE for grading within a 30-day period.

1.4.2.2 CFE manual

The CFE Manual is the text corpus on which all of the questions of the CFE Exam are based. Each question includes a section heading that loosely maps to an

individual section within the manual. However, these sections are rough-grained, often 20–100 pages.

1.5 Code Resources

The code (written in Java [17] and R [18]) for the CFE Agent and its supporting software components are available at https://github.com/jjohnson346/cfe-exam-agent.

CHAPTER 2
Related Work

In this chapter, as we discuss related work, we do so in the context of the question we attempt to address in this dissertation – What balance can we strike between algorithm accuracy, complexity, and reasoning justification in QA? There is likely a trade-off among these competing prioirities. For example, as the accuracy of a system increases, its capability of providing reasoning for its justification likely decreases, in accordance with the no-free-lunch principle. And its complexity likely increases, as well. There are likely a range of possibilities along these competing dimensions, but as of yet, we don't know what they are. We attempt to address that question in this dissertation with our comparative analysis of various algorithms with varying degrees of accuracy, complexity, and justification. We look to the prior work, discussed below, for context.

2.1 Project Aristo

Project Aristo, sponsored by the Allen Institute of Artificial Intelligence, is an ongoing, ambitious research project whose aim is to build an agent that can programmatically acquire knowledge from text such that it can apply reasoning and knowledge to pass 4^{th}-grade science exams [19]. This challenge is designed to push the limits of AI in particular directions – namely in the areas of knowledge, modeling, reasoning, and language. Although shallow information-retrieval-based techniques can be used to successfully answer some questions on the test, others will require incremental advances in these dimensions of AI research [19]. Clark [19] offers a simple ontology of question types that can be described as follows:

2.1.1 Basic Questions

These are questions that require taxonomic knowledge (*is-a* relationships) or terminological (definitional) knowledge. AI systems can commonly answer these successfully using information-retrieval techniques, or using lookup tables.

Example, as given in [19]: The movement of soil by wind and water is called

A. Condensation

B. Evaporation

C. Erosion

D. Friction

2.1.2 Simple Inference Questions

These are questions for which the answer is not explicitly expressed in the text corpus but which must be inferred from a number of pieces of information that are.

Example, as given in [19]: Which example describes an organism taking in nutrients?

A. Dog burying a bone

B. Girl eating an apple

C. An insect crawling on a leaf

D. A boy planting tomatoes in a garden

This example requires knowledge that eating is a process by which an organism takes in nutrients and that an apple contains nutrients.

2.1.3 More Complex World Knowledge

These are questions that require a rich understanding of the world and a linguistic knowledge that can be applied to them to answer the questions.

Example, as given in [19]: Fourth graders are planning a roller-skate race. Which surface would be the best for this race?

A. Gravel

B. Sand

C. Blacktop

D. Grass

This question draws on a number of pieces of knowledge: roller-skating occurs on a surface, skating is fastest on a smooth surface, skating faster is best for a race, and blacktop is smooth, (and other surfaces are *not* smooth) [19].

2.1.4 Diagrams and Mathematics and Geometry

These are question types that are also encountered on the 4^{th} grade science exams, but their nature will not be encountered on the CFE exam[2]. So, these types will not be discussed here.

2.1.5 Progress on Project Aristo

Clark and Balasubramanian [20] discuss some of the issues encountered during this project, and in particular, the tasks of programatic translation of natural language sentences into formal logic and computational reasoning using these programmatically generated assertions.

2.1.5.1 Representing the Meaning of Natural Language Sentences

The approach here is to chunk the natural language sentences in a way similar to a parse of the sentence, breaking up the sentence into noun and verb phrases, and so on. Then, one would manually apply re-write rules by which the chunks can be rearranged into implications. These re-write rules encapsulate syntactic manipulation and domain knowledge.

One problem encountered during this task is the issue of generics, rules that might appear, according to a literal interpretation, to all objects of a particular type in a domain, but from a practical perspective, do not, such as "Dogs chase cats", as explained in [20]. It is fair to say that it is not actually the case that at all points in time if there's a dog it must necessarily be chasing a cat. However, taken literally, this sentence would appear to indicate just that. Therefore, instead of translating this literally into formal logic, first there are probabilities applied to

[2]After a comprehensive review of the questions in the study package for the CFE exam, it was concluded that no questions required knowledge of mathematics or geometry

all of the implications that might arise from this single sentence, where probabilities are assigned based on linguistic rules, as explained in [20].

- Dog ⇒ dog chases cat 0.01
- Cat ⇒ dog chases cat 0.01
- Chase ⇒ dog chases cat 0.01
- Dog chasing ⇒ dog chases cat 0.9
- Chased cat ⇒ dog chases cat 0.8 [19]

2.1.5.2 Reasoning with the Extracted Formalizations

Making inferences of entailed assertions not explicitly stated in the text is the ultimate goal of formalizing the natural language text. However, like the first task, this task also comes with complications. In the case of generics, their representation appears to lead to a downstream problem of having to resolve multiple symbols representing the same object, as explained in [20]. In logic-programming environments, such as Prolog, [21, 22, 23, 24, 25, 26] (mentioned by the author as one of the environments adopted for project Aristo), its Unique Names Assumption means that two symbols cannot be assigned as equivalent. As one approach to this problem, the relation of equality can be developed explicitly, but that means this relation must be worked into every implication, cluttering the knowledge-base. A more workable alternative, as pointed out by the author, is to build into the system the assumption that if an implication rule implies something that appears to repeat another assertion already in the knowledge-base, then assume it actually is already in the knowledge-base, thereby applying a minimization principle in order to avoid duplication of assertions [20].

2.1.6 Perspective on Performance of Project Aristo

The approach of Project Aristo is ambitious – QA using programmatic extraction of knowledge is sparsely charted in AI and attempts by other systems for

whom accuracy and speed are the top prioirities have generally avoided it,[3] This project is still underway, and thus, it remains to be seen what levels of accuracy can be achieved. Even when it does, however, the reports will be with respect to their chosen approach, and does not offer a measure of performance with shallower approaches.

2.2 Halo Project

The Halo Project [27] was a contest sponsored by Vulcan Inc. (founded by Paul Allen), conducted in 2004, in which three teams, SRI, Ontoprise, and Cycorp, set out to create knowledge representation and reasoning systems that could answer test questions similar to those for the Chemistry AP exam based on a text corpus of 50 pages of text and on a sample of 50 multiple choice questions. This test required not only correctness of answers, but also answer justifications that demonstrates the deep reasoning required by the subject matter (of chemistry). So, the systems in this competition needed to address three principal concerns: knowledge representation, QA, and answer justification.

SRI employed an approach that included knowledge engineers and professional chemists (the only team to include such domain experts). They also started with a system that had a relatively large database of real-world knowledge, in which the logic used was called KM logic. Finally, they began the effort using the sample of multiple-choice questions. In the end, this team's results were the best of three, particularly in the area of answer justification.

Ontoprise created a system from scratch (no base-level knowledge-base system) based on a proprietary logic, called F(rame) logic. It required the least runtime, but its accuracy lagged behind SRI's system.

Cycorp built a system using an existing platform, based on a logic called Cycl. This system proved the slowest and least accurate among the three.

[3]Refer to discussion of Watson below, where we mention the tendency to *not* use relation detection for looking up answers to *Jeopardy!* questions.

2.2.1 Perspective on Halo Project Performance

In general, the systems all performed better than expected, averaging around 40% accuracy, equivalent to a score of about AP 3 out of 5 [27]. Cycorp, however, took a long time to execute and its accuracy wasn't quite as good as the other two systems. Justifications were transparent – a natural consequence of the based on knowledge representation and reasoning.

One question for which we don't have an answer is how these systems might have fared on this exam had they been designed to use shallow techniques instead of knowledge-base techniques, where in such a case, reasoning justifications would likely not have been as complete, but accuracy may have been higher. Furthermore, what would have been the implication for relative complexity under a shollow approach?

2.3 Watson

Watson [11] is the famed QA system that beat *Jeopardy!* champions in a matchup in 2010. Watson is built with a pipelined architecture consisting of a number of components, listed briefly below, along with massive parallelization allowing simultaneous computation of multiple algorithms and subsequent selection of the one whose confidence is highest within three seconds. It consists of the following components, as explained in [11]:

2.3.1 Focus and LAT Detection

Lexical answer type (LAT) detection is the determination of words that identify the entity type of the answer, typically from among the words provided in the clue. The focus is the sequence of words in the clue which when replaced by the answer form a valid statement.

2.3.2 Relation Detection

Relation detection is the extraction of semantic relationships from unstructured or semi-structured text. Once the relations have been detected, they are typically re-expressed in a structured format, such as Resource Description Format (RDF). This process is related to information extraction (IR), but in IR, there is

typically a much narrower focus on a closed set of semantic relationships, whereas in relation detection the idea is to cast a wider net for capturing a wide range of relationships. Interestingly, in the case of Watson, although relation detection was used at various points in the QA process, including LAT determination and passage and answer scoring, relation detection was found to be of limited help in the context of direct answer lookups. Indeed, curated databases, in general, were used as a means for looking up answers in less than 2% of questions [11].

2.3.3 Hypothesis Generation

Hypothesis generation involves generating candidate answers by searching through the system's sources, extracting answer-sized snippets from the search results, and plugging those snippets back into the question, thereby forming a hypothesis. During development, the performance goal for this stage in the pipeline was that the correct answer must be present (binary recall) in the top 250 candidates at least 85% of the time. Hundreds of answer candidates are generated by this step.

2.3.4 Soft Filtering

Soft filtering is the lightweight filtering of answer candidates. 'Lightweight' means that this filtering is not computationally intensive, and that more computationally intense forms of filtering will be performed downstream after the answer candidate set has already been whittled down as much as possible. An example of soft filtering is the calculation of the likelihood that an answer candidate has the correct LAT. Filtering thresholds are determined by machine learning.

2.3.5 Evidence Retrieval

Evidence Retrieval is the gathering of evidence for a candidate answer that passes the soft filtering step, including re-initiating a query against the corpus and against the knowledge-base, but with the candidate answer included in the query. This step looks for other passages that include the candidate answer in the context of the original query.

2.3.6 Evidence Scoring

Scoring is done through intensive computational analysis of the candidate answers and evidence, including numerous scoring modules (approximately 50 modules).

2.3.7 Ranking of Hypotheses

Hypothesis ranking uses machines learning, along with mixture of expert techniques to arrive at a final decision on the best answer to the question.

2.3.8 Perspective on Watson Performance

As of the time [11] was published, Watson's performance was considered to be roughly at the same level as the best *Jeopardy!* players, scoring 85% when answering 75% of questions posed to the system, and 67% when answering 100% of posed questions. This level of accuracy was truly groundbreaking in the area of QA, and clearly, the *Jeopardy!* game itself dictated the dual priorities of accuracy and speed as paramount. However, no mention is made as to the level of justification provided for each of these answers. We have to assume that unlike accuracy, justification was not a major priority. What might the accuracy have been had it provided a greater level of justification? This is the question we attempt to answer in this dissertation.

2.4 LASSO QA System

LASSO [29] is the QA system developed by Moldovan et al. at the Natural Language Processing Lab at Southern Methodist University for the TREC-8 competition in 1999. The architecture of this system can be decomposed into three principal components, discussed below, as explained in [29]:

2.4.1 Question Processing Component

The question processing component processes the question posed in natural language, determining the expected answer type, and ultimately, formulating a query to be submitted to the Information-Retrieval engine in the Paragraph Indexing Component (discussed below). The task of question processing breaks down

into the following sub-tasks:

2.4.1.1 Question Type determination

Question type determination is based on a coarse-grained taxonomy of question types, including who, what, where, why, and how, and then by a more fine-grained taxonomy of subtypes. This subcategorization serves as the expected answer type, to be used later during the answer extraction step.

2.4.1.2 Question Focus Determination

Question focus determination is the selection of a sequence of words such that the answer could replace them in the question. For example, as explained in [29], the question, 'What is the largest city in Germany?' is a question where the question focus is, "largest city." Interestingly, it is not always the case that question focus could serve as the sequence of words to use in the query for the IR engine. Consider the question [29]: "In 1990, what day of the week did Christmas fall on? Here, day of the week is the question focus, but does not serve as an effective query for retrieving the docs to answer this question.

2.4.1.3 Question Keywords Selection

This is the process by which the query terms are determined, which is based on a number of heuristics, iteratively applied in order to ultimately develop a query that retrieves promising paragraphs from the document collection likely to contain the answer to the question. These heuristics include the following, as listed in [29]:

- Keyword Heuristic 1: include non-stop words enclosed in quotations
- Keyword Heuristic 2: include named entities recognized as proper nouns
- Keyword Heuristic 3: include all complex nominal and their adjectival modifiers
- Keyword Heuristic 4: include all complex nominal not included in heuristic 3
- Keyword Heuristic 5: include all nouns and noun modifiers.

- Keyword Heuristic 6: include all other nouns recognized in the question
- Keyword Heuristic 7: include all verbs from the question
- Keyword Heuristic 8: include the question focus

2.4.2 Paragraph Indexing Component

In LASSO, the information resources are broken up into paragraph units, which serve as the document unit for indexing in the information-retrieval engine. Paragraph indexing is the technique used in the LASSO system by which the paragraphs are scored to indicate their relative likelihood of containing the correct answer to the question. The paragraph indexing component performs the following two subtasks:

2.4.2.1 Query Execution

This is the step in which the query formulated during question processing is executed against the inverted index by the information retrieval engine. The IR engine component is a Boolean retrieval-based engine that retrieves paragraphs from the document collection based on the query created in the query processing component. Boolean retrieval is used instead of vectorization where cosine similarity is used because boolean retrieval allows for greater recall at the expense of lower precision. The paragraph indexing process serves as the finer-grained filter that prunes the results to a manageable number of paragraphs.

2.4.2.2 Paragraph Indexing

This is the step in which a score is assigned based on a number of statistics, including a same-word-sequence score, distance-score, missing keywords score. This is the strategy for filtering the large number of documents returned by the wide net cast by the IR engine component.

2.4.3 Answer Processing Component

The answer processing component extracts candidate answers from the highest-scoring paragraphs of the paragraph indexing component. Included in this compo-

nent is a parser which provides the part of speech tagging necessary for identifying candidate answers, given the expected answer type. Candidate answers are then scored based on a number of statistics, including same-parse-subtree score, same sentence score, and matched keyword score.

2.5 Answer Type Classifiers

One important aspect of open-domain QA is the task of answer type identification. As explained in the discussions of Watson and LASSO, answer type determination is of great importance in QA systems since proper determination of the answer type allows for greatly reducing the set of candidate answers to the question. Only answers whose features (semantic or syntactic) indicate the same type as that of the question are considered, while other candidates are filtered out. Assuming the answer type of the question is correct, (not a trivial assumption), the QA system is in a much better position to narrow down the set toward the final, correct response.

Li and Roth [30] describe a machine-learning-based question classifier which is hierarchical in nature, where at the high level, there are 6 coarse-grained categories (abbreviation, entity, description, human, location, and numeric value), and at the low level, there are up to 50 fine-grained categories. For example, for the entity type, subtypes include animal, body, color, creative, currency, and so on, while for location, subtypes include city, country, mountain, state, and other. Their approach leverages machine learning in which feature types are analyzed, as opposed to specific features, allowing the training to extrapolate to a much larger feature set without having to annotate as large a training set. Primitive features include words, part-of-speech (POS) tags, chunks (non-overlapping phrases), named entities, and head chunks (the first noun-chunk in a sentence). Higher order feature types are derived from these, i.e., classes of features that are automatically extracted for a given answer type category.

2.5.1 Perspective on LASSO Performance

Performance of this system was measured based on accuracy, showing results of accuracy scores were 55.5% for short answer and 64.5% for long answer questions [29]. Again, we are left asking ourselves the question, had the engineers used different techniques which would have yielded more transparent justifications, how much would cost would be imposted in terms of accuracy?

2.6 Prismatic

Prismatic [28] is a lexicalized relation resource that programmatically processes a large corpora (30 GB) in order to identify and store the ways entity and relation information are represented in unstructured text. The system works by breaking its processing into three phases: corpus processing, frame extraction, and frame cut extraction. Corpus processing is primarily concerned with parsing the text using IBM's slot-grammar parser. Frame extraction uses the parse information to break up the text into frames, where a frame is a text representation of a group of entities and the relations between them. Prismatic produced 1 billion frames from the 30 GB text corpus. Lastly, frame cut extraction is a process in which the system extracts small parse sequences from the frames (e.g., S-V-O (subject-verb-object), or (N-P-O (noun-preposition-object)) in order to analyze recurring patterns of text within them. The redundancy of terms across these frame cuts allows the system to conclude extensional information (relations between specific instances of entities, such as the relation, located-in(Troy, New York)) and intensional information (e.g., *located-in(city, state)*). This sort of information has applications in entity type identification, relation extraction, and question-answer.

2.6.1 Perspective on Prismatic

This system was not a QA system. It was an entity detection system that was used in Watson at various points in the QA process, as mentioned above. It is relevant to the work here as it helps us appreciate the extensive engineering that went into the development of shallow relation detection. But even with that investment, relations in curated databases were used seldomly for directly answering questions

in a lookup fashion.

2.7 Semantic Approaches in the Fraud Domain

This section highlights work using a formal semantic approach to activities related to fraud/fraud detection. The work summarized below inspired Version 4 of the CFE Agent, covered in Chapter 6.

2.7.1 The Lying Machine

Clark's doctoral dissertation [31] discussses the foundations and development of a prototype lying machine, a system which leverages a theory of mind in order to identify and exploit cognitive frailties in human reasoning based on mental heuristics and biases in order to assert plausible falsehoods that deceive its human audience. This work lays the psychological and logico-computational groundwork for machine-generated mendacity.

We mention this work even though it is not directly involved in QA in the same sense that the systems discussed above are because it involves deception (inherent in fraud) treated in a logico-computational framework.

2.7.2 Formal Definition of Fraud

Firozabadi [32] discusses a formal definition of fraud for the purposes of verifying electronic trade procedures, such as those inherent in business-to-business electonic commerce. This work breaks down trade activities into two subgroups – primary actions and control actions where control actions are designed to prevent deception in the execution of primary actions – and expresses these activities using formal logic, using modal operators for action, belief (doxastic logic component), and obligation (deontic logic component). We also mention this work for the same reason we mention [31].

2.7.3 Perspective on Related Work

The systems above are all tremendous contributions to the field of QA and to AI, in general. But the question that remains is: "How might these systems have

fared in either of their respective problem domains had they been designed with different priorities in mind?" How much would accuracy would have been sacrificed had the Watson provided its user with reasoning justifications for its answers? Would the Project Halo systems performed significantly better had a knowledge-base approach *not* been used? Could similar scores have been earned using less complex algorithms? What are the costs of sacrificing accuracy in favor of justifications, and vice versa? This is the question on which we hope to shed some light as we look at the various versions of the CFE Agent, where we implement algorithms with different mixes of priorities, and then apply them to the *same problem domain*. By comparing their scores on a battery of CFE Exam questions, assessing their complexity, and characterizing their levels of justification, we can obtain a sense of the options on the landscape of possibilities for balancing the competing priorities of accuracy, complexity, and reasoning justification.

CHAPTER 3
Version 1 – A Shallow Agent

This chapter describes the development, algorithms, and performance of Version 1 of the CFE Agent which uses strictly shallow natural language processing (NLP) techniques [2]. It should be emphasized that these algorithms are simplistic; some exploit structural patterns of the exam questions while others rely on bag of words techniques, in order to observe how well the agent performs with relatively little true intelligence. Performance measurements indicate this agent performs surprisingly well. Some of the algorithms based on question structure, for example, reveal strong correlations with correct answers, as will be discussed in detail below, although the agent does not perform at a passing level. Also, the agent provides very little justification for its answers, and thus, there is still ample motivation for a more sophisticated approach.

3.1 Resource Procurement and Customization

The first phase of development was the conversion of the study package downloaded from the ACFE website to a form that could be processed mechanistically. The study package includes the CFE Manual in pdf format, practice questions embedded within custom software, and a practice exam, also consisting of questions embedded within custom software. It was necessary to convert each of these components into text-formatted documents for the CFE Agent.

The first of these components, the CFE Manual, is a 2,500 page pdf document that serves as the principal resource upon which all of the exam questions are based. Conversion of this document involved an extensive amount of work for two reasons: First, the pdf document itself was password-protected, rendering most pdf conversion programs useless. And second, the few software packages that *were* capable of bypassing the password protection produced intermittent errors in the generated

Portions of this chapter previously appeared as: J. Johnson, "Toward an intelligent agent for fraud detection – the CFE Agent," in *Deceptive and Counter-Deceptive Machines: Papers from the AAAI Fall Symposium*, no. FS-15-03. AAAI Press, 2015, pp. 21–26

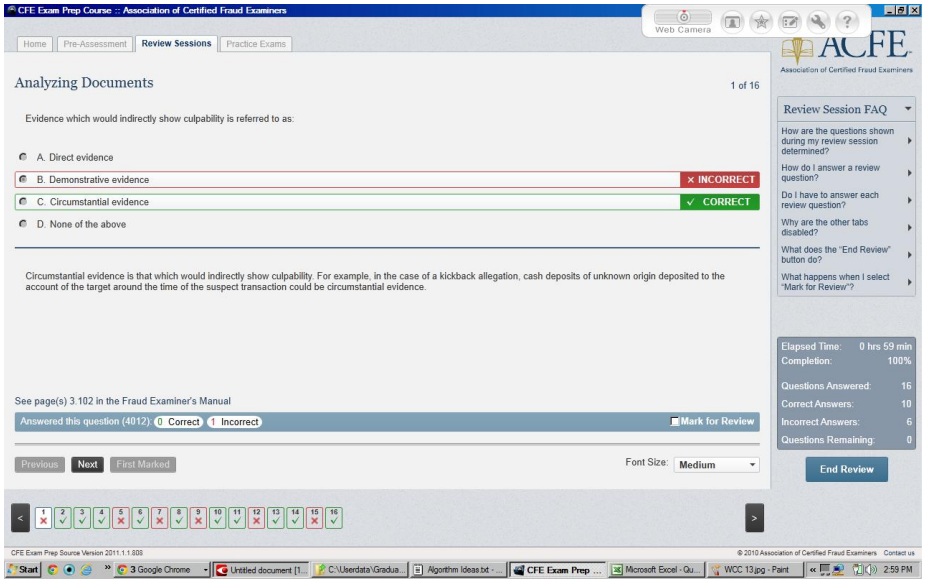


Figure 3.1: Embedded Practice Question

text, inserting spaces randomly between the letters of words. Given the large volume of content, a program employing various parsing techniques was written to sequentially walk through the text, detect errors, make corrections automatically based on certain rules, or prompt for correction if an error was detected but manner of correction was uncertain. Although a significant amount of effort was expended to develop this technology, in the end, much of the document was edited manually.

The other main component of the prep package is the database of 1,500 questions embedded in the study- package software. The extraction of these questions from the software was an involved, multi-step process. First, a screenshot file, as shown in Fig. 3.1 [34], was created for each question. Second, optical character recognition (OCR) software was used to convert the graphic screen shot files into

```
* CFE Exam Prep Course :: Association of Certified Fraud Examiners
,©
Web Camera
B00
Pre-Assessment Review Sessions Practice Exams
Analyzing Documents
Evidence which would indirectly show culpability is referred to as: C A. Direct
evidence
1 of 16
C B. Demonstrative evidence
C C. Circumstantial evidence
x INCORRECT
C D. None of the above
Circumstantial evidence is that which would indirectly show culpability. For example,
in the case of a kickback allegation, cash deposits of unknown origin deposited to the
account of the target around the time of the suspect transaction could be
circumstantial evidence.
See page(s) 3.102 in the Fraud Examiner's Manual
Answered this question (4012):QQBB5) {MQZE3D
■I Mark for Review
Previous I Next
Font Size:
Medium
1
X
2 II 3 II 4 □ M □
```

Figure 3.2: Converted, Unscrubbed Practice Question

text files, as shown in Fig. 3.2. Unfortunately, the important data needed for each question was intermingled among nonsensical overhead text in each of these text files. So, a custom program was created to parse these text files, remove the unnecessary data, and decompose each question into its component parts — topic, question stem, options, correct answer, and explanation — as shown in Fig. 3.3. Finally, these files were manually reviewed and edited to ensure proper format for the agent. The product of this process was a collection of 1,500 text files, one for each practice exam question, and each containing name-value pairs for the relevant question fields named above.

```
Section=Analyzing Documents

Stem=Evidence which would indirectly show culpability is referred to as:

Options=Direct evidence | Demonstrative evidence | Circumstantial evidencex | None of the
above

CorrectResponse=2

Explanation=Circumstantial evidence is that which would indirectly show culpability. For
example, in the case of a kickback allegation, cash deposits of unknown origin deposited
to the account of the target around the time of the suspect transaction could be
circumstantial evidence.See page(s) 3.102 in the Fraud Examiner's Manual
```

Figure 3.3: Scrubbed Practice Question

3.2 CFE Agent Design

The CFE Agent is an object-oriented program written in Java. It takes as input a collection of question files and produces as output a sequence of answers which are, in turn, compared against the corresponding correct answers in order to calculate a score. There are five principal elements in the design of the CFE Agent: the CFE manual, the question profile, the algorithm set, algorithm profile data, and the algorithm-selection algorithm.

3.2.1 Question Profile

After having reviewed the entire database of questions in the study package, it was determined that there are certain characteristics certain questions share in common. This was thought to be a natural way of partitioning the questions according to a crude ontology, each class of which could serve as a means for optimizing an algorithm for the questions of that class. For example, one of the most obvious of these characteristics is whether the question is a true-false question or a multiple-

choice question. And if the question is true-false, does it include any of the following terms: “always,” “never,” “must,” or “only”? On the other hand, if a question is multiple-choice, is its last option, “All of the above”? Some other criteria discovered that are worth noting are whether the question includes options with more than four words in it, whether the question contains the word ‘except’, and whether the question’s last option is ‘None of the above’.

3.2.2 Algorithm Set

Various algorithms were developed and incorporated into the model. The system currently employs nine custom algorithms that utilize very shallow natural language processing techniques. Descriptions of a few example algorithms are given in the next three sections:

3.2.2.1 Max Joint Probability Algorithm

This algorithm considers only the possible answers, treating each as a sequence of words whose joint probability is to be calculated. It calculates this joint probability for each option by finding the probability of each word among the words of the manual section from which the question was drawn (indicated by topic) and by applying the simplifying assumption of independence between words. The joint probability, then, is simply calculated as the product of the probabilities of the individual words of the option, normalized for the number of words (so that short answers are not over-weighted). The algorithm selects the option with maximum joint probability.

3.2.2.2 Max Frequency Algorithm

The Max Frequency Algorithm simply counts up the total number of times each answer option is encountered in the manual section from which the question was drawn and selects the option whose tally is the largest. Unlike the Max Joint Probability Algorithm, the frequencies are based on occurrences of entire phrases, not on the counts of individual words within them.

3.2.2.3 False Select

This algorithm applies only to true-false questions, simply selecting false, always. Remarkably, this algorithm proved extremely effective for such questions that have "must," "always," "only," or "never" in the stem, achieving an astonishing 78.7% accuracy rate on this class of questions (see Fig. 3.12). The remarkable success of such a simple algorithm is an indication of the effectiveness of gaming strategies on certain questions, but also an indication of weaknesses in the way certain questions on this exam are designed.

3.2.3 Algorithm Profile Data

The algorithm profile data is a table showing the accuracy each algorithm on each question question profile, tabulated from running every algorithm on every question in a training set of 1,300 question selected from the battery of 1,500 questions. For each of these questions, a profiler component performed the following steps: First, determine the question profile. Second, execute each algorithm on the question, recording whether each algorithm produced the correct answer. Finally, summarize the results. A portion of this summary data is shown in Fig. 3.4.

3.2.4 The Selection Algorithm

The selection algorithm is the procedure for selecting the proper algorithm to apply to a given question. It is based on the algorithm profile data described above. For a given question in the 200-question test set, the CFE Agent chooses the algorithm with the highest accuracy rate for questions in the training set with the same profile.

3.3 CFE Agent Demo

This section outlines a walkthrough of the execution of the CFE Agent for a very short exam, consisting of only four questions. Despite its short length, it should provide the reader with a good sense of the basic mechanics of the agent.

Fig. 3.5 shows the startup of the agent. The name of a configuration file is passed in as a command-line parameter to the program. This file provides a

Index	Trials	ALL_ABOVE	TRUE_SELECT	FALSE_SELECT	MAX_FREQ	MAX_FREQ_PLUS	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM
0	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	177	0.000	0.000	0.000	0.294	0.294	0.311	0.316	0.316	0.305
2	223	0.000	0.000	0.000	0.413	0.413	0.269	0.359	0.413	0.300
3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	375	0.000	0.587	0.413	0.000	0.000	0.000	0.000	0.000	0.533
9	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
11	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
12	57	0.000	0.228	0.772	0.000	0.000	0.000	0.000	0.000	0.368
13	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
14	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
15	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
16	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
17	32	0.000	0.000	0.000	0.125	0.125	0.188	0.094	0.094	0.188
18	12	0.000	0.000	0.000	0.083	0.083	0.000	0.083	0.083	0.083
19	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
21	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
22	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
23	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
25	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
27	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
28	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
29	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
30	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Figure 3.4: Training Set Profile Data

number of settings for the execution environment, including the mode (interactive vs. batch), the name of the exam file, and log detail level. In this demo, the agent is executing with full detail so that it is easier to understand what is happening under the covers for each question.

The detail logging shows that one of the first things the CFE Agent does when it starts up is construct an internal representation of the CFE Manual. This internal representation is in the form of a graph, each of whose nodes represents a section of the document. More specifically, the graph is a tree whose root node represents the entire manual, which in turn possesses links to child nodes, each representing one of the four main sections of the manual (Financial Transactions and Fraud Schemes, Law, Investigation, and Fraud Prevention and Deterrence) each of which in turn possesses child nodes representing sub-sections to those sections, and so on. Each

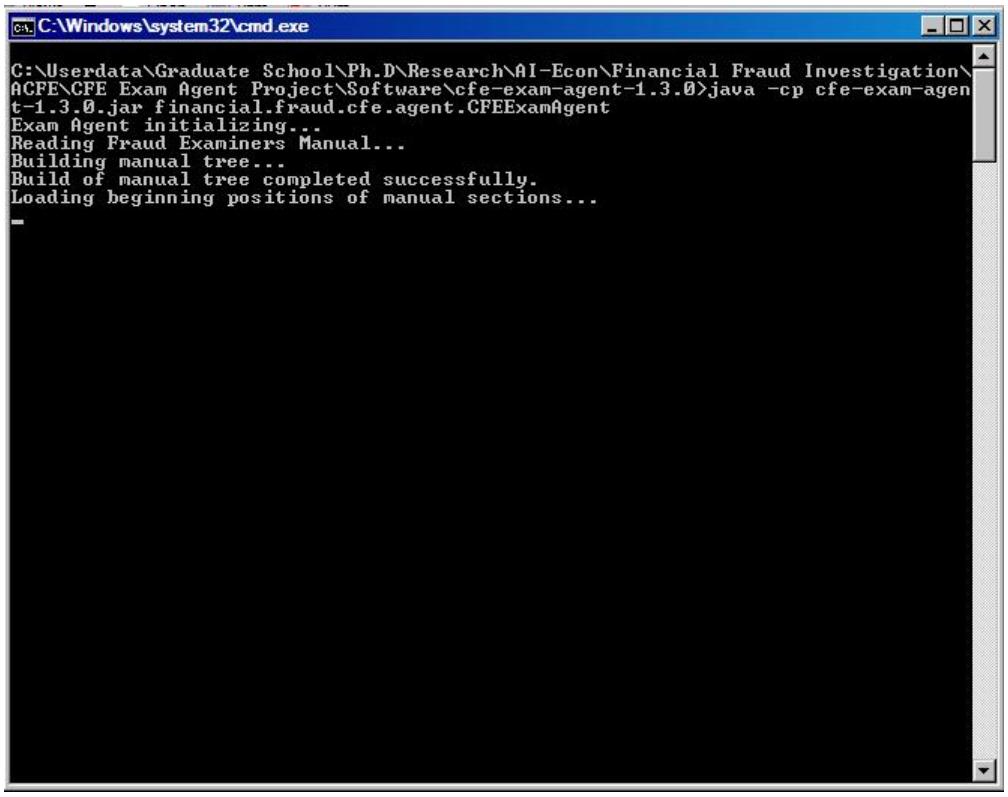


Figure 3.5: CFE Agent Startup

node stores information about the section it represents in addition to the text itself, including a hash table for word counts, sub-subheadings, et cetera. This information facilitates the text-analytic computations the agent performs as it answers questions on the exam. It should be noted that much of the basis of this tree data structure is based on the items listed in the manual's table of contents, and for even finer-grained detail nodes, on text features within the corpus itself (e.g., lines of text consisting of all capital letters terminating in carriage return generally denote a sub-section of text). Finally, in addition to the tree data structure, the internal representation of the manual includes additional data structures mapping the text of the manual for optimizing access to particular manual subsections.

In Fig. 3.6, the screen shows the completion of the startup sequence of the CFE Agent. First, the CFE Agent gives a line-by-line report showing the nodes loaded into the tree data structure. Then, it shows some additional configuration information showing the exam file, execution mode, and runtime status. At this

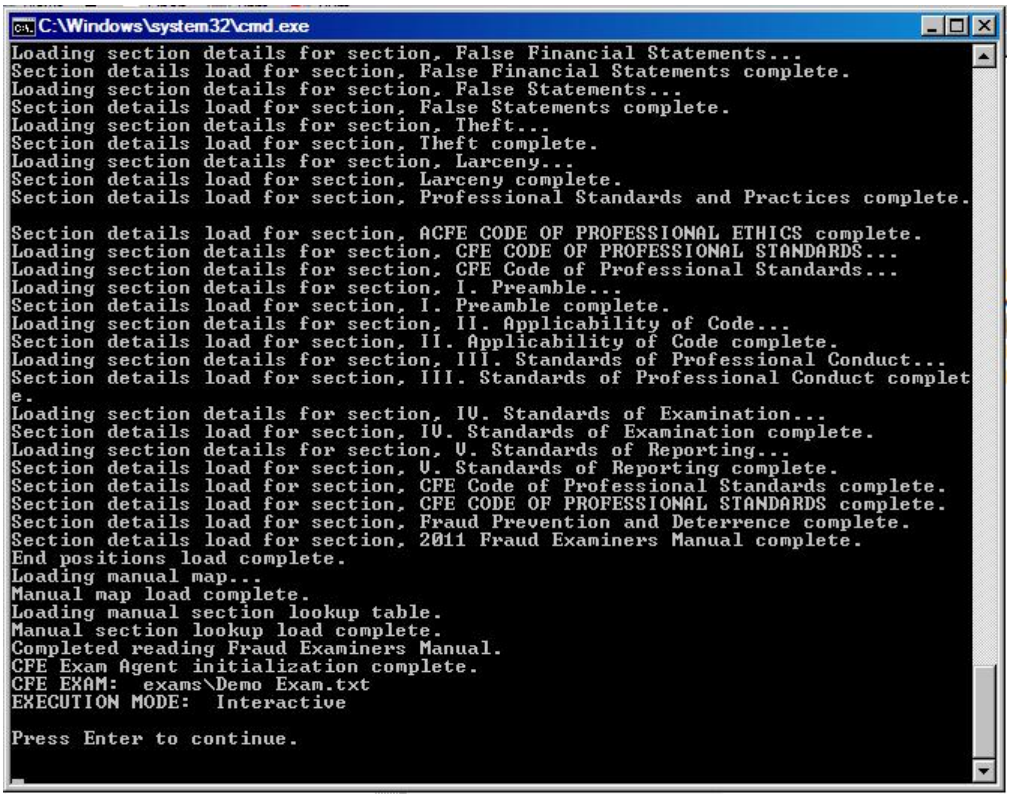


Figure 3.6: CFE Agent Startup Completed

point, the CFE Agent is ready to begin the exam.

In Fig. 3.7, the first question of the exam is shown along with four possible answers. After the user presses return, the CFE Agent selects the optimal algorithm for that particular question and executes it.

Fig. 3.8 shows log detail for the CFE Agent as it attempts to choose the correct answer for the first question. First, as described in the discussion of the question profile, the agent makes an assignment to a question-profile category based on some key features of the question (whether it's a true-false question, a question at least one of whose options contains more than 4 words, et cetera). Once it assesses the question profile, it chooses an algorithm which based on prior experience exhibits maximum accuracy on questions having the same profile. For this question, the profile assignment is category 2, meaning the question has been categorized as a "definition" question, essentially one which attempts to test for knowledge of domain-specific vocabulary. The experience data of the agent indicates the best-

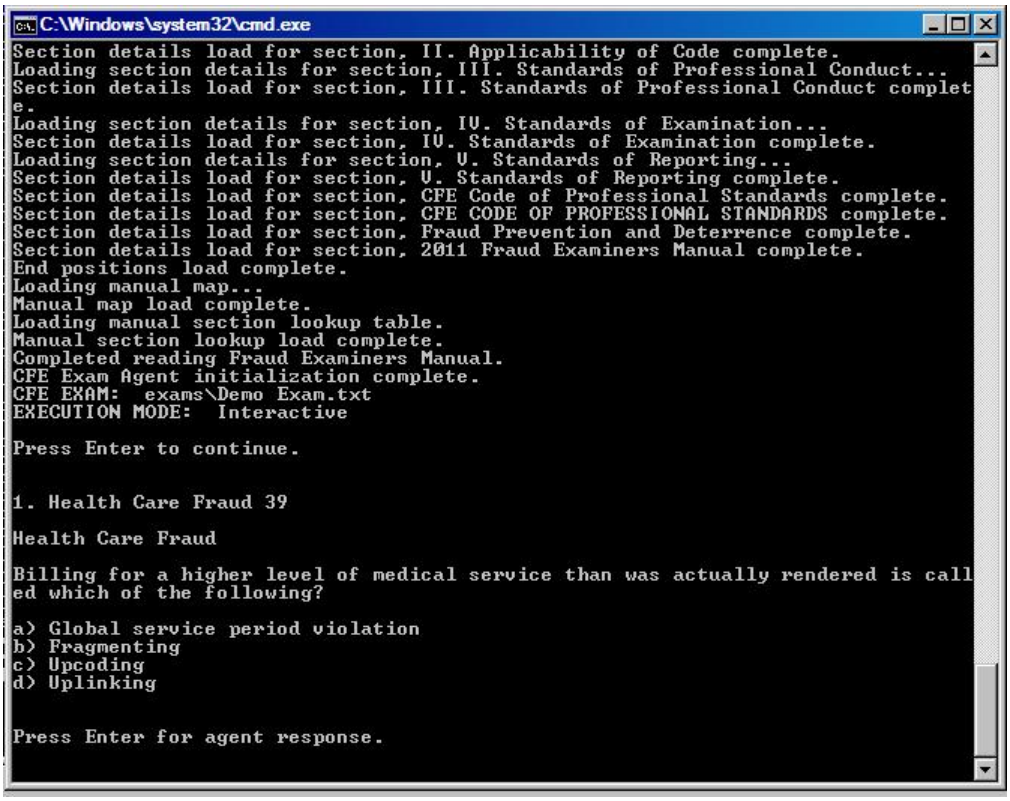


Figure 3.7: First Question

performing algorithm is the Max Frequency algorithm, which when applied for this question results in the selection of choice c, the correct answer.

Fig. 3.9 shows the CFE Agent answer the second question. This question appears to be similar to the first in that it also earns a category assignment of 2, meaning it is another "definition" question. By the same experience-based reasoning as in the first question, the agent applies the Max Frequency algorithm and selects answer a, the correct answer.

Fig. 3.10 shows an attempt at the third question. However, this time the agent is not successful at choosing the right answer. Again, it assigns this question to the same category as the two prior questions and uses the same Max Frequency algorithm, but it is led astray by the fact that the fourth option, "cost," is a common word whose frequency in the section corpus is over-weighted relative to the other options for this question. This is an example where the current level of sophistication of the agent is insufficient to correctly answer questions where one of the options

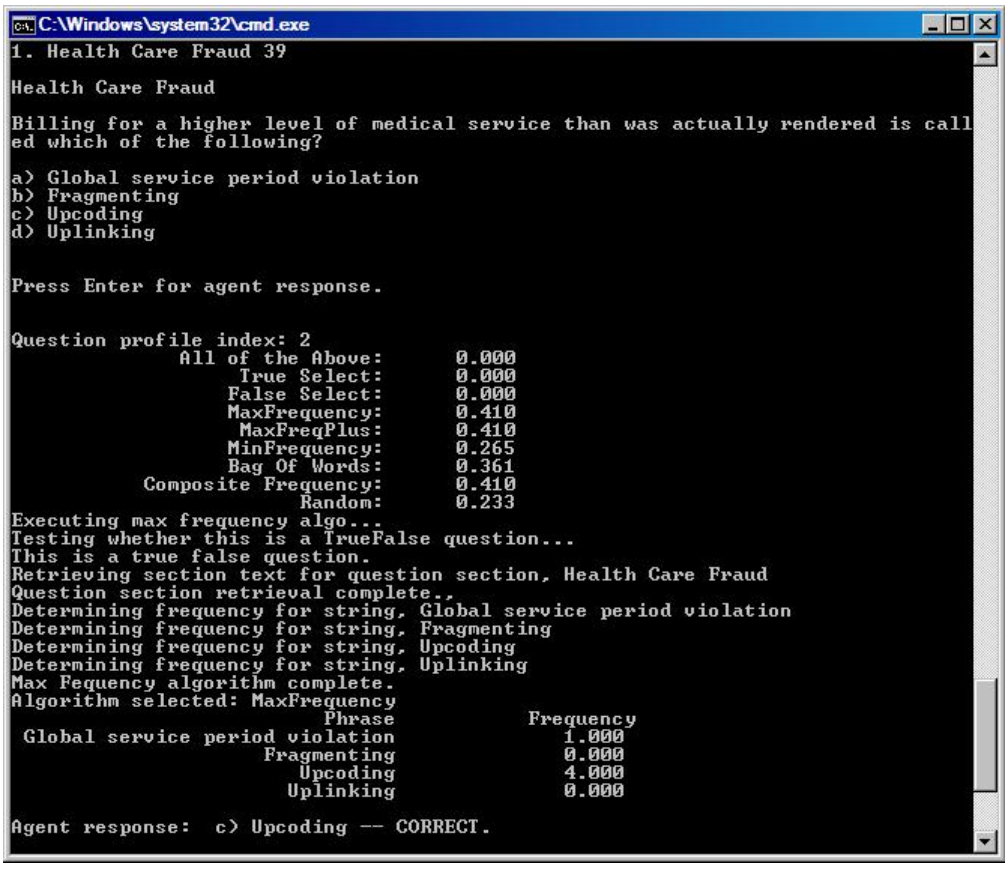


Figure 3.8: First Question Completed

may consist of a word or phrase that is over-represented in the corpus. Further work is needed to develop the natural language processing algorithm to compensate for this over-weighting, while preserving its relative success on other questions in the same category.

Fig. 3.11 shows the final question, which is one of a different type – an "All of the Above" question. Here, the CFE Agent chooses an algorithm appropriately called 'All of the Above' because it simply selects "All of the Above" whenever its an option. (As shown in the data table, this algorithm has a remarkable 86% success rate (see Fig. 3.14), lending one to question certain aspects of the CFE Exams design, as mentioned earlier.) The agent applies this algorithm and gets the answer correct. Finally, the agent terminates.

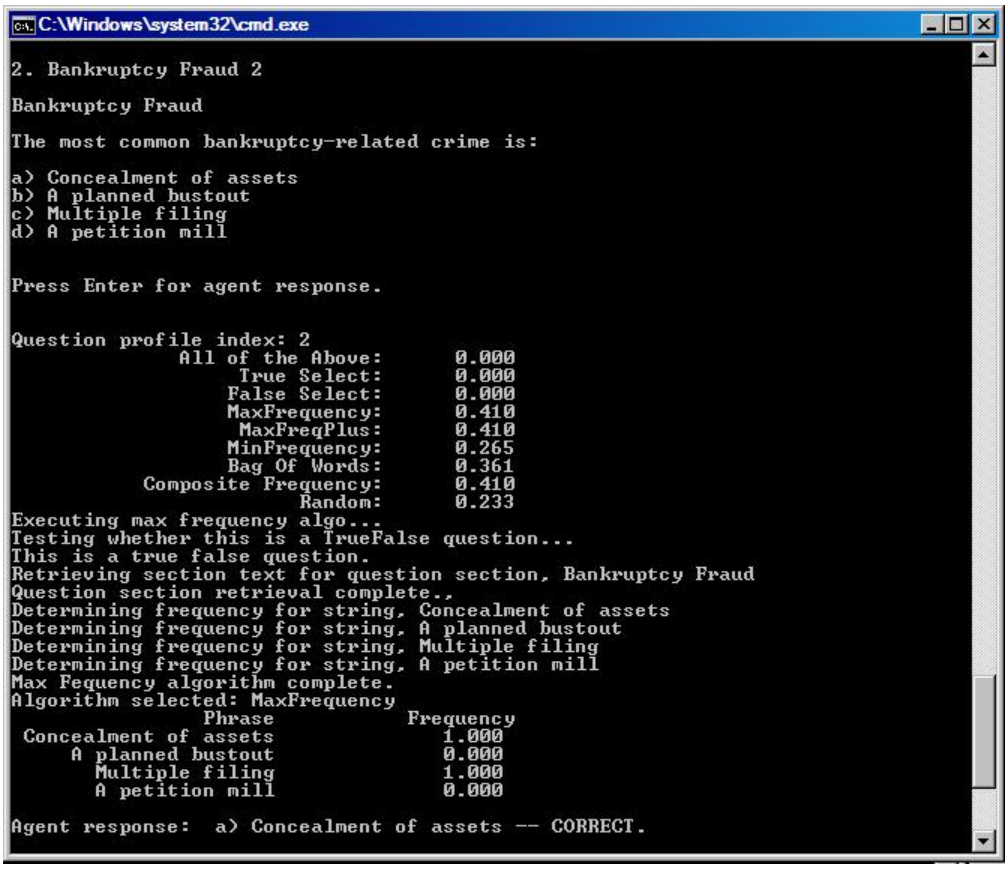


Figure 3.9: Second Question

3.4 CFE Agent Results - A Good Start

A detailed look at the performance of the agent on the entire battery of 1,500 questions, assuming the agent employs the most accurate algorithm for each question, given its profile, shows the agent demonstrates promising, statistically significant performance. Consider the question of whether the agent actually performed any better than random guessing. For true-false questions, random guessing would be expected to yield an accuracy rate on the of approximately 50%. (Given there are a relatively large number of true-false questions (506), it is reasonable to assume the experimental accuracy rate should be very close to this theoretical value.) Likewise, for multiple-choice questions with four options, random guessing should yield an accuracy rate of approximately 25%. (Again, for the same reason as for true-false questions, the experimental accuracy should be close to the theoretical expected value.) However, the empirical data shown in Fig. 3.12 and Fig. 3.14 indicates that

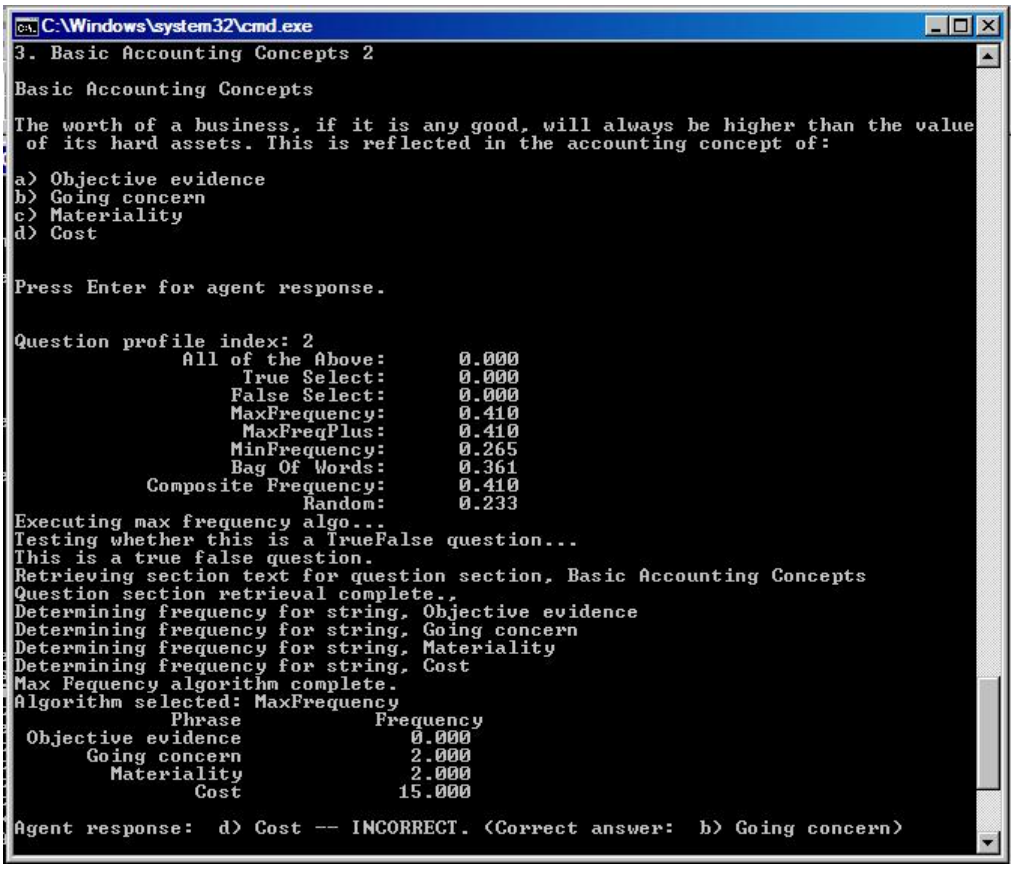


Figure 3.10: Third Question

for true-false questions, the observed accuracy rate is over 58% on this training set, giving a z-score of over 3.729 (the z-score for a p-value of 1% is 2.575), and that for multiple-choice questions, the accuracy rate is just over 47.8%, giving a z-score of 16.67. For both classes of questions, then, the null hypothesis of performance consistent with random guessing is rejected by a large margin at the 99% significance level.

The above analysis covers the entire battery of 1,500 questions. However, in developing our agent, this battery was broken into two subsets: a training set consisting of 1,300 questions and a test set consisting of 200 questions. The idea here is to utilize the training set to determine the optimum algorithm for each profile (that is, the one whose accuracy is highest), and then to apply that knowledge on the test set by applying the best algorithm for each question, based on its profile. The process of breaking the collection of questions up into a training set and test

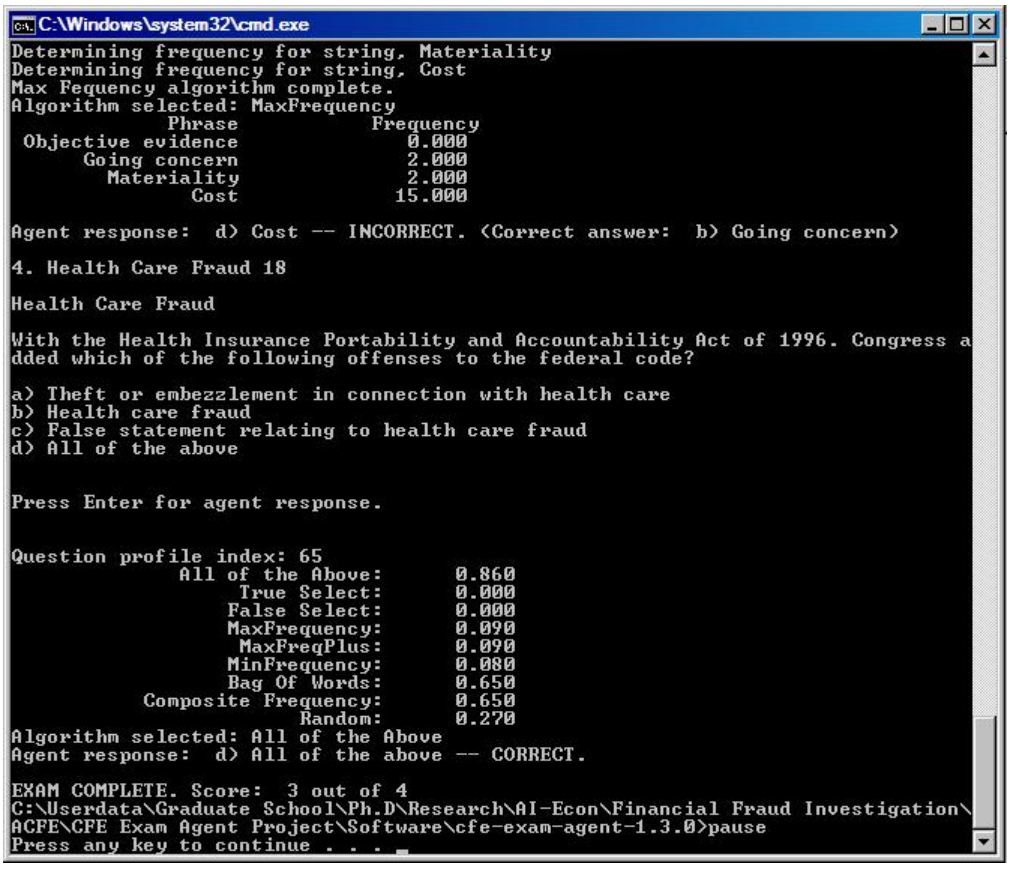


Figure 3.11: Final Question and Termination

True/False

Index	Trials	ALL_ABOVE	TRUE_SELEC	FALSE_SELEC	MAX_FREQ	MAX_FREQ_	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM	AGENT	PROFILE DESCRIPTION
8	459	0	0.562	0.438	0	0	0	0	0	0.521	0.562	true/false
12	47	0	0.213	0.787	0	0	0	0	0	0.574	0.787	true/false/absolute
	506										0.58289921	

Figure 3.12: Results for True-False Questions

set was done via a programmatic procedure that selected a random, proportionate selection of questions for each CFE-exam subtopic. Fig. 3.16 shows that the results for the training set are comparable to those for the 1,500-question battery, as should be expected. Using this data, the CFE Agent was executed on the 200-question test set, where it answered 98 correct, or 49%. Although this is short of passing, it's an encouraging start, and provides a good baseline for subsequent versions of the agent.

Finally, it should also be pointed out that during development of later versions

True/False

H0: Agent answer rate = 0.5
H1: Agent answer rate > 0.5

Question Count	506
E(Question Score)	0.5
Var(Question Score)	0.25
E(Exam Score)	0.5
Var(Exam Score)	0.00049407
Std(Exam Score)	0.02222771
Observed Exam Score	0.58289921
z-score for Observed Exam Score	3.72954321
z-score for p-value of 1%	2.575
Conclusion:	Reject H0 in favor of H1.

Figure 3.13: Hypothesis Test for True-False Questions

Multiple Choice

Index	Trials	ALL_ABOVE	TRUE_SELEC	FALSE_SELEC	MAX_FREQ	MAX_FREQ_	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM	AGENT	PROFILE DESCRIPTION
1	206	0	0	0	0.282	0.282	0.296	0.296	0.296	0.277	0.296	long options
2	249	0	0	0	0.406	0.406	0.265	0.365	0.406	0.301	0.406	definition
17	37	0	0	0	0.135	0.135	0.189	0.108	0.108	0.162	0.189	except/long options
18	12	0	0	0	0.083	0.083	0	0.083	0.083	0.417	0.417	except/definition
33	101	0	0	0	0.416	0.228	0.347	0.287	0.287	0.257	0.416	none of above/long options
34	202	0	0	0	0.53	0.51	0.322	0.386	0.51	0.243	0.53	none of above/definition
49	2	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	none of above/except/long option
65	100	0.86	0	0	0.09	0.09	0.08	0.64	0.64	0.21	0.86	all of above/long options
66	82	0.78	0	0	0.073	0.085	0.122	0.598	0.085	0.268	0.78	all of above/definition
81	2	0	0	0	0	0	0	0	0	0.5	0.5	all of above/except/has long options
82	1	0	0	0	1	1	1	0	1	1	1	all of above/except/definition
	994										0.47897686	

Figure 3.14: Results for Multiple-Choice Questions

of the agent, it was noticed that an important question feature had been overlooked in the original analysis: the notion of a "not" question. An example of a not question is: "Which of the following is NOT one of the major standards of generally accepted accounting principles?". It was found to be necessary to break out questions of this variety from other questions that were otherwise of the same type, but did not include the 'NOT' term. Thus, the analysis was completed once more on the training set, showing the following results in Fig. 3.17. It should be noted from this table, for example, that among the 223 questions formerly categorized as "definition" questions, 40 are now categorized as "definition/not" questions. The same phenomenon occurs for other categories, as well.

The performance measures given above show us that simplistic techniques,

Multiple Choice

H0: Agent answer rate = 0.25
H1: Agent answer rate > 0.25

Question Count	994
E(Question Score)	0.25
Var(Question Score)	0.1875
E(Exam Score)	0.25
Var(Exam Score)	0.00018863
Std(Exam Score)	0.01373433
Observed Exam Score	0.47897686
z-score for Observed Exam Score	16.6718638
z-score for p-value of 1%	2.575
Conclusion:	Reject H0 in favor of H1.

Figure 3.15: Hypothesis Test for Multiple-Choice Questions

True/False

Index	Trials	ALL_ABOVE	TRUE_SELEC	FALSE_SELEC	MAX_FREQ	MAX_FREQ_	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM	AGENT	PROFILE DESCRIPTION
8	375	0	0.587	0.413	0	0	0	0	0	0.533	0.587	true/false
12	57	0	0.228	0.772	0	0	0	0	0	0.368	0.772	true/false/absolute
	432										0.6114097	

Multiple Choice

Index	Trials	ALL_ABOVE	TRUE_SELEC	FALSE_SELEC	MAX_FREQ	MAX_FREQ_	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM	AGENT	PROFILE DESCRIPTION
1	177	0	0	0	0.294	0.294	0.311	0.316	0.316	0.305	0.316	long options
2	223	0	0	0	0.413	0.413	0.269	0.359	0.413	0.3	0.413	definition
17	32	0	0	0	0.125	0.125	0.188	0.094	0.094	0.188	0.188	except/long options
18	12	0	0	0	0.083	0.083	0	0.083	0.083	0.083	0.083	except/definition
33	91	0	0	0	0.407	0.242	0.341	0.297	0.297	0.209	0.407	none of above/long options
34	169	0	0	0	0.55	0.527	0.308	0.343	0.527	0.266	0.55	none of above/definition
49	2	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	none of above/except/long option
65	89	0.843	0	0	0.101	0.101	0.09	0.618	0.618	0.18	0.843	all of above/long options
66	70	0.743	0	0	0.086	0.1	0.143	0.586	0.1	0.329	0.743	all of above/definition
81	2	0	0	0	0	0	0	0	0	0	0	all of above/except/has long options
82	1	0	0	0	1	1	1	0	1	1	1	all of above/except/definition
	868										0.4770357	

Figure 3.16: Results for Multiple-Choice and True-False Questions on 1300-Question Training Set

True-False:

Index	Trials	ALL_ABOVE	TRUE_SELECT	FALSE_SELECT	MAX_FREQ	MAX_FREQ_PLUS	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM	Agent	Description
16	375	0.000	0.587	0.413	0.000	0.000	0.000	0.000	0.000	0.483	0.587	true-false
24	57	0.000	0.228	0.772	0.000	0.000	0.000	0.000	0.000	0.544	0.772	true-false/absolute
Total Count:	432										0.611	

Multiple Choice:

Index	Trials	ALL_ABOVE	TRUE_SELECT	FALSE_SELECT	MAX_FREQ	MAX_FREQ_PLUS	MIN_FREQ	B_OF_W	COMP_FREQ	RANDOM	Agent	Description
1	177	0.000	0.000	0.000	0.294	0.294	0.311	0.316	0.316	0.243	0.316	long-options
4	196	0.000	0.000	0.000	0.454	0.454	0.265	0.393	0.454	0.260	0.454	def
6	27	0.000	0.000	0.000	0.111	0.111	0.296	0.111	0.111	0.259	0.296	def/not
33	32	0.000	0.000	0.000	0.125	0.125	0.188	0.094	0.094	0.313	0.313	except/long-options
36	12	0.000	0.000	0.000	0.083	0.083	0.000	0.083	0.083	0.250	0.250	except/def
65	91	0.000	0.000	0.000	0.407	0.242	0.341	0.297	0.297	0.209	0.407	none-above/long-options
68	169	0.000	0.000	0.000	0.550	0.527	0.308	0.343	0.527	0.260	0.550	none-above/def
97	2	0.000	0.000	0.000	0.500	0.500	0.500	0.500	0.500	0.000	0.500	none-above/except/long-options
129	89	0.843	0.000	0.000	0.101	0.101	0.090	0.618	0.618	0.270	0.843	all-above/long-options
132	64	0.813	0.000	0.000	0.094	0.109	0.078	0.641	0.109	0.250	0.813	all-above/def
134	6	0.000	0.000	0.000	0.000	0.000	0.833	0.000	0.000	0.500	0.833	all-above/def/not
161	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.500	all-above/except/long-options
164	1	0.000	0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	all-above/except/def
Total Count:	868										0.497	

Figure 3.17: Training Set Results with NOT Feature Breakout

though shallow and lacking reasoning justification, can actually produce decent results. So, as we look at more sophisticated/complex techniques, we seek to measure the marginal benefit that we gain in terms of accuracy and justification.

CHAPTER 4
Version 2 – A More Intelligent Agent

In this chapter, we describe Version 2 of the CFE Agent, enhanced with improved algorithms using information retrieval for locating the relevant section(s) of the CFE Manual for each question.[4] Version 1 of the CFE Agent is limited to coarse-grained sections as defined by the question sections of the CFE Exam and Manual. Unfortunately, these sections of the manual are effectively so large, it's difficult for the agent to drill down to the text *directly* related to a given question. For intellectual-property-theft questions, for example, the "Theft of Intellectual Property" section of the manual is 89 pages long. For any question on this topic, pulling the entire section of the CFE manual will assure that the answer sought lies somewhere within our target document. But with 89 pages of text in the section, we might still get the answer wrong. Version 2 refines the filtering algorithm to help avoid this problem.

This chapter is laid out as follows: First, we discuss the method for breaking up the CFE manual into fine grained sections, leveraging the table of contents and text metadata. Next, we explore Lucene [35], an Apache software [36] component that provides information retrieval functionality, which was used in the development of Version 2. Next, we briefly describe some support tools there were built to facilitate algorithm analysis and refinement in this iteration. And finally, we discuss the algorithms of Version 2 and their performance.

One note before proceeding: The terms, "training set" and "test set" are used throughout this chapter in reference to the battery of questions against which the algorithms discussed below were applied. Indeed, we talk repeatedly about testing our algorithms against the training set, specifically. It is important to note here that "training set" is a term of convenience in this chapter, and not one that should be taken to suggest a set for which we've optimized parameters vis à vis machine learning. In fact, the algorithms we discuss in this chapter involve no training, per se, and so, it is perfectly valid to measure or test the performance of each of these

[4]For an overview of the topic of Information Retrieval, see Appendix A.

algorithms against the "training set".[5] Lastly, it should be mentioned that at the end of this chapter, we use the performance of each algorithm on the questions of the training set to determine the optimal algorithm to use on each question in the test set in order to measure overall performance of Version 2 of the agent.

Finally, before moving any further, during the development of Version 2, a deeper review of the definition-type questions was in order. Some questions were re-classified according to finer grained criteria, including the following: Nine questions were reclassified as "I, II, III, and IV" type. Six questions for which "Any of the Above" was an option were re-classified as "All of the Above" type. There were 26 questions whose answers weren't derived from the Fraud Examiners Manual but instead from a different text, The Corporate Fraud Handbook. Finally, five questions were re-classified as "NOT" questions. After reclassifying these questions, we were left with 150 definition questions in our training set upon which to base our algorithm performance analysis, reduced from our original 196 questions used for Version 1.

4.1 Transforming the CFE Manual into a Document Collection

The CFE Manual is structured as a text book. As such, it is structured hierarchically, as most textbooks are, complete with features embedded in the text that make this hierarchy apparent. The most obvious feature is the table of contents (TOC) and headings embedded in the text for each of the chapters, sections, subsections, and so on. In fact, the CFE Manual has a number of tables of contents, including a main table of contents, at the front of the manual, and a set of area-specific TOCs - one for each of the major test areas - Financial Transactions and Fraud Schemes, Law, Investigation, and Fraud Prevention and Deterrence. Fig. 4.1 shows a section of the TOC relating to Financial Statement Analysis, a topic contained in the area of Financial Transactions and Fraud Schemes. The summary TOC combined with the area-specific TOCs combined with text features (capitalized sub-sub-sub section titles within the text itself) were all used to programmatically break

[5] In chapter 5, however, we *do* employ machine learning, and the training set and test set are used according to their conventional definition.

up the manual into a hierarchical structure of documents. Fig. 4.2 shows a portion of this document structure, where on the left we see the Bankruptcy Fraud subsection, a subtopic of Financial Transactions and Fraud Schemes, and the breakdown of documents, each of which named according to a numeric identifier and a title corresponding to a title for the subsection, along with indentation showing its level in the document tree. On the right side, we see the contents of one of the documents covering the subtopic of Bankruptcy Court. Notice that in each document we have not only the text of the section but also a a title field (Bankruptcy Court), a question section field (Financial Transactions and Fraud Schemes), a path giving the sequence of section titles starting from the root node of the CFE manual hierarchy to the current document, and finally, the stemmed contents of the document, using the Porter Stemmer algorithm. All of these elements were compiled for each document as possible inputs to algorithms developed downstream for answering questions.

4.2 Lucene – A Tool for Information Retrieval

Lucene [35] is a highly popular Apache software product that implements IR using a combination of Boolean Retrieval and the Vector Space Model (VSM) [37, 38, 39, 40]. First, it narrows the document set using boolean retrieval, and then it ranks the remaining documents using VSM. The algorithms described below were implemented using this tool.

However, before implementing any algorithms, the document collection into which the CFE manual was decomposed was indexed using the Lucene software. Indexing is the process of essentially creating files containing the underlying data stuctures required by Lucene's combined boolean retrieval/VSM retrieval algorithm, including an inverted index for the collection and document vectorization data. In fact, a Lucene index was created for the documents of each question section. For a given question, Lucene processes a query against the index for the corresponding question section, returning a set of documents from which an answer is drawn. When implementing the QA algorithms discussed below, the first step to any of these algorithms was locating the proper index within which to search for relevant documents, based on the exam section/question section that corresponds to the

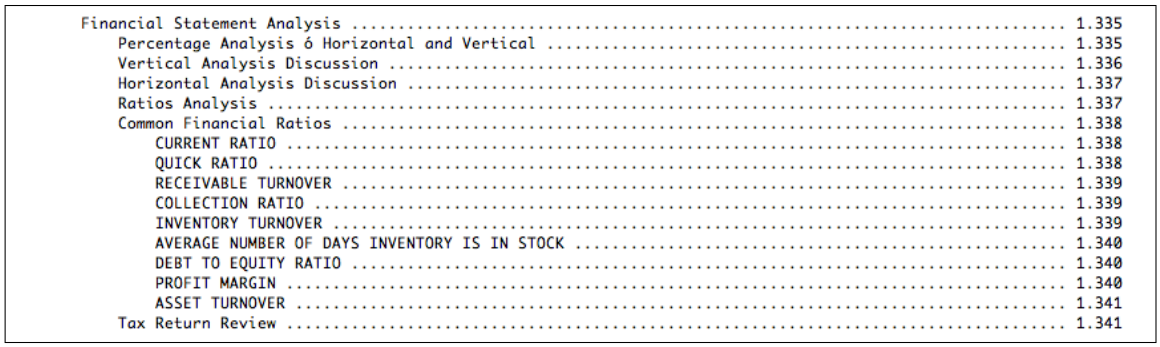


Figure 4.1: A section of the table of contents from the CFE Manual

question at hand. Fig 4.3 shows a portion of the Lucene indexes created by this process. Notice that for each question section within each exam section, there are three binary files created by the Lucene indexer component that contain the inverted index and document vectorization information required for the query processing component to be used in the algorithms discussed below.

One other thing of note is that as part of this process, the contents field and the title field were stemmed according to the Porter Stemmer algorithm. Stemming is a form of semantic normalization, where words offering different senses of the same semantic unit are transformed so that they are treated as equivalent during the document scoring computation in the IR process. For example, different words for run - run, ran, running - are all considered semantically equivalent as a result of stemming.

4.3 Analysis Tools for Algorithm Development

As outlined in prior chapters, the goal of the CFE agent is to answer questions correctly while providing justification for those answers. As algorithms were developed toward this end, and in particular, as we attempted to refine the accuracy of the agent by making its search functionality more fine-grained, it was determined early in the process that one of the most critical pieces of information was to understand

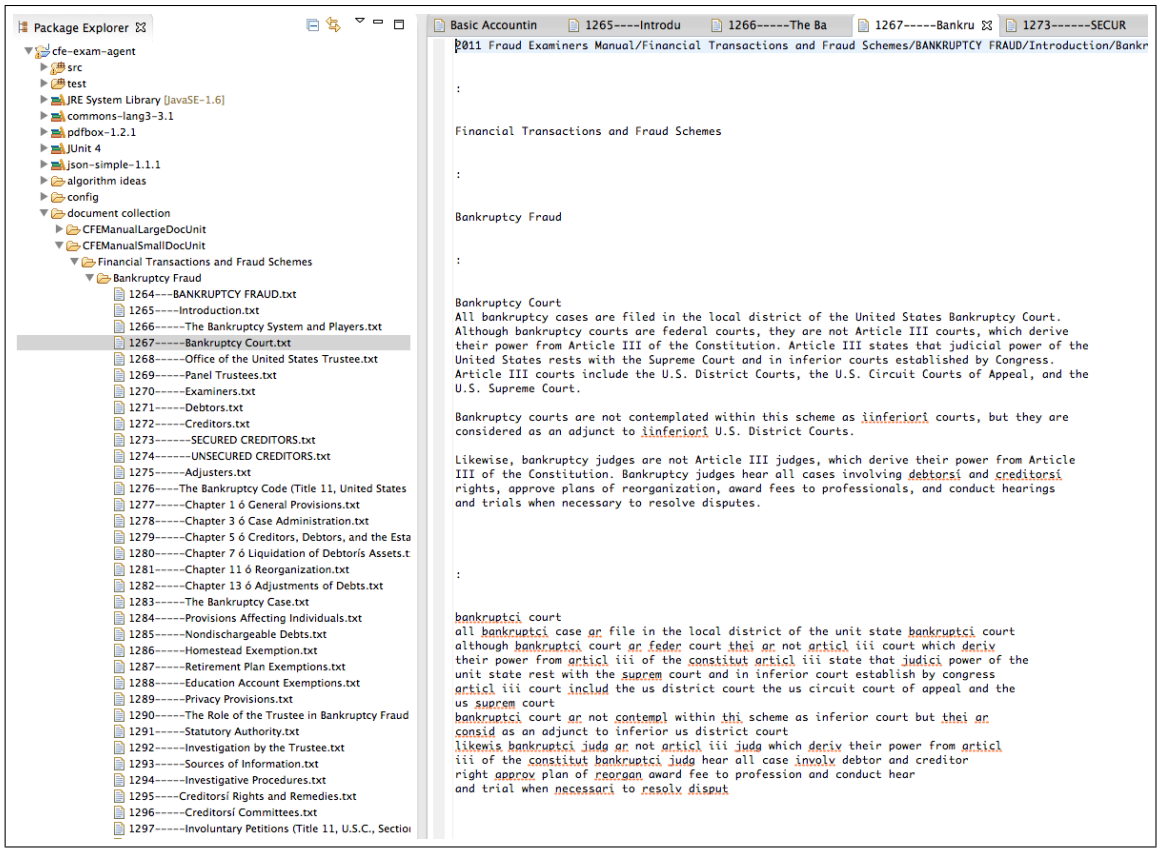


Figure 4.2: The CFE Manual as a Document Collection

how to target each type of question - What features for a given type of question could be exploited when searching for an answer? Specifically, how does the answer present itself in the manual to a question of a given type? Is it contained in a single document, or multiple documents? Are terms found in the options commonly found in the contents of the document, or are they found in the title? Depending on the answer, how often is that the case? Is it always true, or only sometimes? Tools that aided in this investigation were critical to the development of "smarter" algorithms.

At a macro level, the profiler component, developed for Version 1 of the agent provided an initial analysis tool. As discussed, it supplied a breakdown of question by macro-features, as well as the success rate of the initial algorithms created for

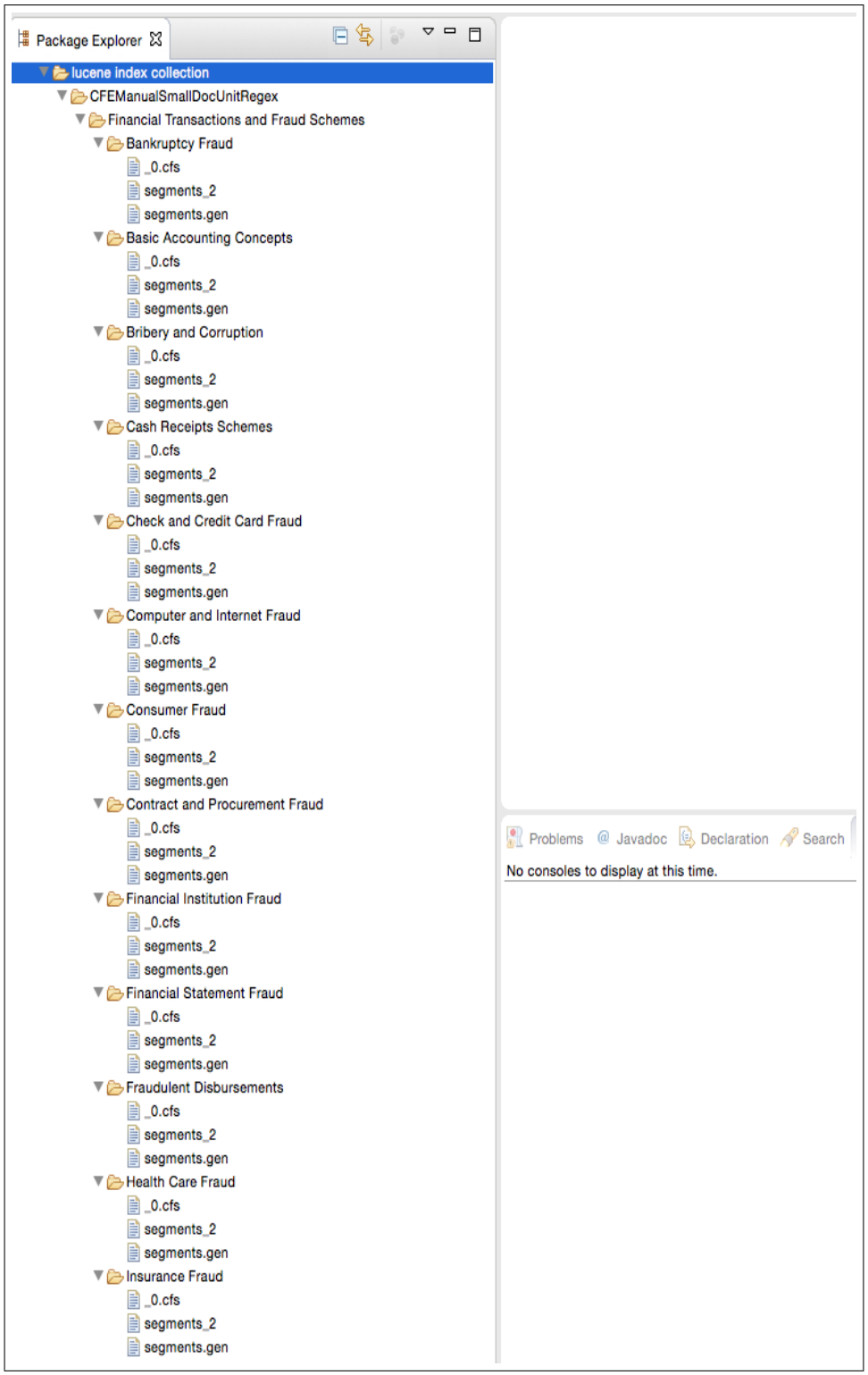


Figure 4.3: Lucene Indexes for a Portion of the Question Sections

that version. And here in Version 2, the profiler would be used again. However, analysis at a greater level of detail was needed.

One component, called the Question Server, was created to at least partially meet this need. It poses to the agent only questions of a single profile, (all definition questions, or long answer questions, and so on). This provided a means for zeroing in on each question type in isolation, which was key in algorithm development. Fig. 4.4 shows output from the Question Server for definition/NOT questions, questions whose options are a small number of words (and are thus, typically relate to a definition of a concept), and contain the term, "not" in the stem, implying the task is to identify the "odd man out" among the options.

A second component, called the Algorithm Tester, was created on the shoulders of the Question Server, which would demonstate the behavior of each algorithm as it was applied to each question of a given type. This, too, was instrumental in algorithm development.

4.4 Algorithms for Version 2

This section describes the algorithms implemented for Version 2 of the CFE Agent. These algorithms rely heavily on IR, as implemented in Lucene. They also tend to each target a specific question type, in particular, the definition questions - that is, those questions in which each of the options is only a short phrase (consisting of four words or less), and which thus are thought to be most likely the kind of question in which the stem contains a phrase that defines a given concept and the examinee must choose the correct concept from among the four options to which that phrase corresponds. Given that a large portion of the questions in the training and test sets are some form of this type, more refined algorithms that target this type of question appeared to be a wise choice for where to focus our development efforts.

4.4.1 Concept Match Version 1

Concept Match Version 1 leverages IR on both the question stem and on each of the options in order to determine the one that fits best with the stem. The essence

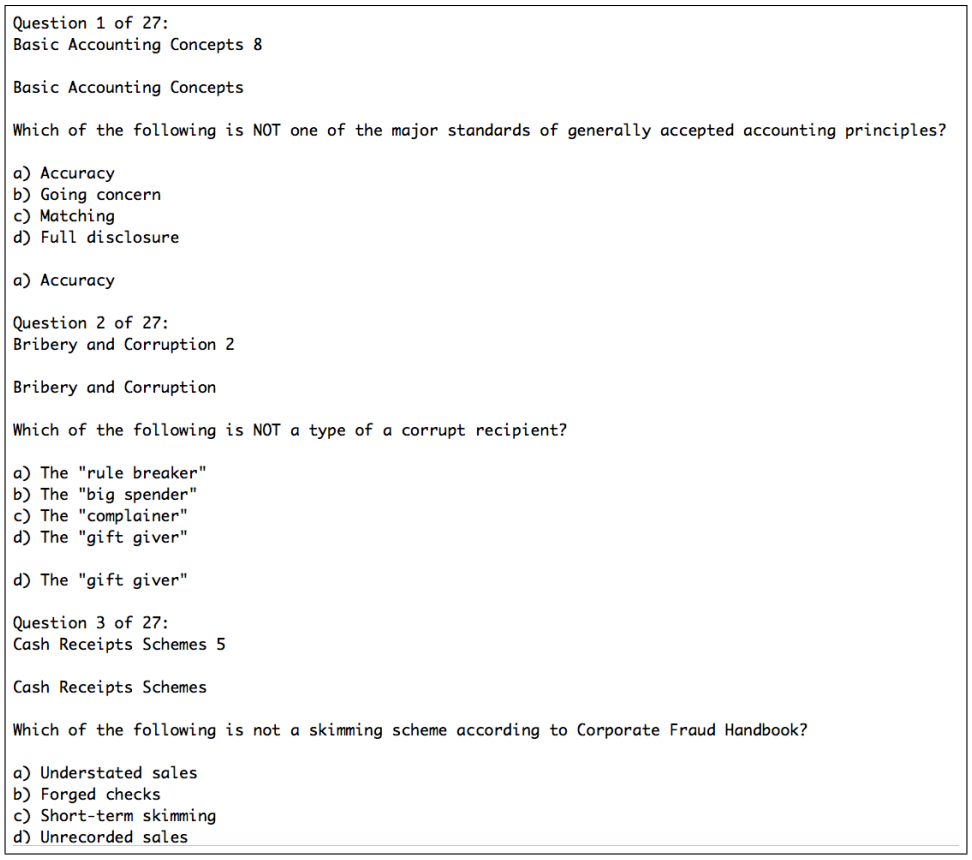


Figure 4.4: Question Server Component Targeting Definition/NOT Questions

of the algorithm is to first, conduct an IR query against the document collection based on the terms of the question stem, then, conduct a query based on the terms of each of the question options, and finally, select the option whose return set has the "best overlap" with the return set for the question stem. How do we define "best overlap"? For the purpose of this algorithm, a return set with the "best overlap" is the return set whose intersection with the stem query return set contains the document with the highest score relative to the intersection sets for other return sets.

Consider an example. Fig. 4.5 shows a Bankruptcy Fraud definition question.

(Although the criterion for being a definition question has only the naive requirement that all options be no more than 4 words, this example demonstrates how this criterion is often sufficient for properly categorizing this type of question, since this question really *is* a definition-type question.) The justification by the agent shows its reasoning - First, it shows the return set for each option of the four options in the question, including for each document its title, id (automatically assigned by the Lucene indexing component), and score. (This example is particularly convenient as each option has the simplifying characteristic of at most one return document in its return set. This is not always the case.) Next, the agent shows the return set for the query based on its stem, "a person who holds a perfected security interest against a person filing bankruptcy". The return set is sorted by decreasing score. Now, the option, "Secured creditor" has a result set that includes the "SECURED CREDITOR" document (it's capitalized because that's the way it appears in the text for the manual), which possesses a higher score in the question stem return set than any of the other documents in the return sets for the other options. (In this particular case, it's also *the* high scoring document in the question stem return set, although this fact is not a requirement of the algorithm.) This means that the "best overlap" is the "SECURED CREDITOR" document, and as a result, the agent picks option, "a) Secured creditor", the correct answer.

Some further details about the mechanics of this algorithm should also be mentioned here. The algorithm conducts a query against the document collection based on the stem to return a collection of 10 documents (this number chosen arbitrarily) relating to the question. This query is conducted against the *contents* field of each document. (Note that as alluded to previously, each document consists of four fields - a title field, a contents field, a question section field, and a path field.) This means that when retrieving documents in response to the query, the terms in the contents field, exclusively, are used to determine the relevance of the document to that query. The document's title and path are not considered (at least not in this algorithm). It should also be noted that before conducting the query, functional phrases including "is referred to as", "are referred to as", "which of the following", and so on, were removed from the stem as they do not offer semantic

information about the nature of the question. Also, the words of the question stem were stemmed just as the terms in the contents field of each document were during the indexing process. This is necessary in order for the scoring to work properly.

As for the queries based on each of the question options, they are executed against the *title* field of each of the documents. So, in order for a document to be found to match well to an option, it must have one or more terms in its *title* in common with those of the option. This greatly narrows the range of documents that produce high ranked hits, and this was by design. It was noticed that on more than a few questions, the options map nicely to subsections whose titles align closely with the options. Unfortunately, however, in the event the question options do not have sibling sections in the manual, this algorithm does not perform well.

4.4.1.1 Agent Justification for Selected Answer

Notice, in the above example, that the agent shows its reasoning for its argument. By giving the scored return set for the question stem, and for each option, the agent provides the detail for its argument for the option it selects. It is clear from the detailed output that among the documents in the returns set for the question stem query and among the set of documents in the return sets for the question options queries, there are two documents that appear in both - SECURED CREDITORS, (id = 9) and UNSECURED CREDITORS (id = 10). Since the SECURED CREDITORS document earns a higher vectorization match score of 0.3375, the reasoning for selecting the Secured creditors, option a, is clear.

4.4.1.2 Concept Match V1 Performance

Finally, we look at the performance of the Concept Match Version 1 algorithm. Fig. 4.6 shows output from the algorithm tester tool described above, showing that among the 150 questions in the training set that were classified as strictly definition questions,[6] this algorithm correctly answered 56 questions, or 37.3% – not an outstanding rate. In fact it is lower than the rate of the Version 1's Max Frequency

[6]There were other questions that met the criterion for the definition question type, but were classified in different but related categories, such as the definition/not category, or definition/except category.

```
Bankruptcy Fraud

A person who holds a perfected security interest against a person filing bankruptcy is referred to as which of the following?

a) Secured creditor
b) Judgment debtor
c) Judgment creditor
d) Unsecured creditor

Doc results for: Secured creditor
1. SECURED CREDITORS(9)  0.9063726

Doc results for: Judgment debtor
1. Chapter 7  Liquidation of Debtors Assets(16)  0.54382354

Doc results for: Judgment creditor
** no docs returned **

Doc results for: Unsecured creditor
1. UNSECURED CREDITORS(10)  0.9063726

conceptDocs: {16=1, 9=0, 10=3}
Stem (lower case): a person who holds a perfected security interest against a person filing bankruptcy

Doc results for stem: a person who holds a perfected security interest against a person filing bankruptcy
1. SECURED CREDITORS(9)  0.33575153
2. Paragraph 7  Fraudulent Transfer or Concealment(44)  0.3160012
3. UNSECURED CREDITORS(10)  0.2941953
4. Knowing Disregard of Bankruptcy Law or Rule(49)  0.23285939
5. Adverse Interest and Conduct of Officers(48)  0.18710557
6. Paragraph 4  False Claims(41)  0.17282711
7. Embezzlement Against the Estate(47)  0.1723838
8. Paragraph 5  Fraudulent Receipt of Property(42)  0.15566334
9. Paragraph 8  Fraudulent Destruction or Alteration of Documents(45)  0.13475572
10. Paragraph 9  Fraudulent Withholding of Documents(46)  0.12726296

Option selected: a) Secured creditor
Correct!
```

Figure 4.5: Concept Match V1 Example

```
Total questions answered: 150
Number question correctly answered: 56(0.37333333333333335)
number of questions where option selected is -1 (no selection): 46
```

Figure 4.6: Performance of Concept Match V1 on Definition Questions

algorithm[7] on the same questions. However, these results also show that for 46 questions this algorithm produced no answer at all. This was because the laser-focused question option queries against the title field of the documents in some cases returned 0 documents for *all* options, thereby causing the algorithm to fail. Fig. 4.7 shows an example of this scenario. With no documents in the set of return sets for the option queries, the algorithm has no other choice than to return -1, (for no option selected). In the next algorithm, Concept Match V2, however, this issue is addressed.

4.4.2 Concept Match Version 2

Concept Match Version 2 extends Concept Match V1 by addressing two major concerns. The first is the situation outlined above in which for the query options no documents are returned because of the tight focus of the queries on the title field. In Concept Match V2, if this scenario occurs, the options returns sets are rebuilt, but instead of searching on the title fields, the the search is performed on the contents fields. This loosens the focus of the option queries, promoting the likelihood of non-empty return sets.

Fig. 4.7 shows shows another question for which Concept Match V1 fails due to empty return sets for the option queries. The two figures, Fig. 4.8 and Fig. 4.9, show the output for Concept Match V2 for the same question. Whereas the V1 algorithm gives up and returns -1, V2 redoubles its effort and re-issues the same option queries against the contents field, resulting in the correct answer.

[7]See Fig. 4.27, row for profile, 4.

```
Financial Statement Fraud 9

Financial Statement Fraud

Revenue is recognized on long-term construction contracts under one of two methods. Which of the following is one of those methods?

a) Disbursement/recovery method
b) Partial-contract method
c) Percentage-of-completion method
d) Cost-to-market method

Doc results for: Disbursement/recovery method
** no docs returned **

Doc results for: Partial-contract method
** no docs returned **

Doc results for: Percentage-of-completion method
** no docs returned **

Doc results for: Cost-to-market method
** no docs returned **

conceptDocs: {}
Stem (lower case): revenue is recognized on long-term construction contracts under one of two methods.  is one of those methods

Doc results for stem: revenue is recognized on long-term construction contracts under one of two methods.  is one of those methods
1. LongTerm Contracts(16)  0.46679574
2. Improper Asset Valuation(21)  0.19583732
3. DEBT TO EQUITY RATIO(62)  0.1853602
4. Financial Statement Fraud Schemes(5)  0.12198833
5. Capitalized Expenses(36)  0.11294544
6. Trends in Financial Statement Fraud(4)  0.10464896
7. Premature Revenue Recognition(11)  0.10460665
8. Percentage Analysis  Horizontal and Vertical(51)  0.08904613
9. BOOKING FICTITIOUS ASSETS(28)  0.085955516
10. Concealed Liabilities and Expenses(34)  0.07611202

option selected: -1
Incorrect.  Correct answer: Percentage-of-completion method
```

Figure 4.7: Concept Match V1: An Example Where No Docs Are Returned for Options

The second major concern this algorithm addresses is demonstrated by the following example shown in Fig. 4.10. The highest scoring document common to both the question stem query return set and option query return sets, (and importantly, also the document containing the correct answer) is docID, 38, "The Business Profile Analysis". Unfortunately, this document is also in the return sets for *two* options – option b, "Preparing the business profile", and option c, "Preparing the vertical analysis", making the correct choice ambiguous. But since Concept Match V1 loads these documents into the conceptDocs hash map[8] in order by option, docID, 38, is

[8]The conceptDocs hash map is used by the concept match algorithms to find the highest scoring

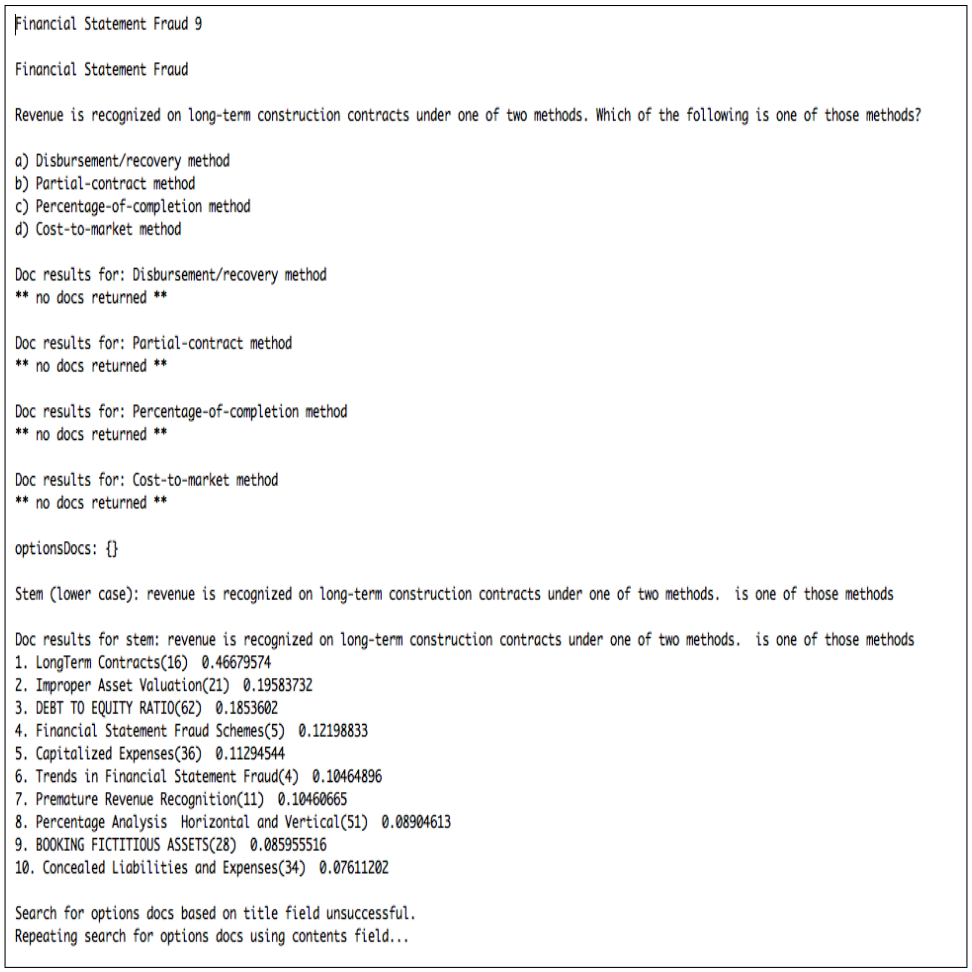


Figure 4.8: Concept Match V2: Fixing the No Docs in Option Queries Return Sets Problem, Part 1

initially mapped by the algorithm to option b, the correct answer, but is *subsequenty* mapped to option c, the incorrect answer. We can see this association in the display

---

document common to both the stem query result set and the option query result sets. This hash map contains a collection of key/value pairs, where the key stores a docID and the value stores an option id, (0, 1, 2, or 3). For a given docID/option pair, the docID is the document ID for a document found in the return set for one of the options, whose id is given in the value field. The line in the output that starts with "conceptDocs:" shows the contents of this map. In Fig. 4.10, we see that docID, 49, is returned for option 1 (b), docID, 3, is returned for option 1 (b), docID, 38, is returned for option 2 (c), and so on.

```
Doc results for: Disbursement/recovery method
1. LongTerm Contracts(16)  0.10989416
2. Concealed Liabilities and Expenses(34)  0.049146157
3. Tax Return Review(65)  0.049146157
4. Fictitious Revenues(6)  0.035104398
5. Multiple Deliverables(17)  0.035104398
6. Percentage Analysis  Horizontal and Vertical(51)  0.035104398
7. Ratios Analysis(54)  0.028083518
8. LiabilityExpense Omissions(35)  0.017552199

Doc results for: Partial-contract method
1. LongTerm Contracts(16)  0.12732214
2. Concealed Liabilities and Expenses(34)  0.05694019
3. Tax Return Review(65)  0.05694019
4. Fictitious Revenues(6)  0.040671565
5. Multiple Deliverables(17)  0.040671565
6. Percentage Analysis  Horizontal and Vertical(51)  0.040671565
7. Ratios Analysis(54)  0.03253725
8. LiabilityExpense Omissions(35)  0.020335782

Doc results for: Percentage-of-completion method
1. LongTerm Contracts(16)  1.5838112
2. Concealed Liabilities and Expenses(34)  0.06884125
3. Tax Return Review(65)  0.06884125
4. Fictitious Revenues(6)  0.049172323
5. Multiple Deliverables(17)  0.049172323
6. Percentage Analysis  Horizontal and Vertical(51)  0.049172323
7. Ratios Analysis(54)  0.03933786
8. LiabilityExpense Omissions(35)  0.024586162

Doc results for: Cost-to-market method
1. LongTerm Contracts(16)  0.18363643
2. Improper Asset Valuation(21)  0.1624789
3. Inventory Valuation(22)  0.1624789
4. Concealed Liabilities and Expenses(34)  0.0821247
5. Tax Return Review(65)  0.0821247
6. Fictitious Revenues(6)  0.058660503
7. Multiple Deliverables(17)  0.058660503
8. Percentage Analysis  Horizontal and Vertical(51)  0.058660503
9. Ratios Analysis(54)  0.046928402
10. LiabilityExpense Omissions(35)  0.029330252

optionsDocs: {16=2, 65=3, 17=3, 34=3, 51=3, 35=3, 21=3, 6=3, 54=3, 22=3}

Options doc with best match: LongTerm Contracts(16)
Option selected: c) Percentage-of-completion method
Correct!
```

Figure 4.9: Concept Match V2: Fixing the No Docs in Option Queries Return Sets Problem, Part 2

of the contents of the conceptDocs hash map, in which the key/value association, 38=2, signifies that docID, 38, is associated with option 2 (that is, option c).[9] As a result, the agent gets this question wrong.

Concept Match Version 2 corrects this problem by recognizing a situation in which a document is included in multiple option return sets. In this case, the algorithm maps the document to the option for which that document earned the highest

[9]Option ids are 0-based, so option 0 corresponds to option a, 1 to b, 2 to c, and 3 to d.

```
Bribery and Corruption 17

Bribery and Corruption

in proving corrupt payments, the fraud examination often begins with which of the following?

a) Interviewing the target
b) Preparing the business profile
c) Preparing the vertical analysis
d) Interviewing of the co-conspirator

Doc results for: Interviewing the target
** no docs returned **

Doc results for: Preparing the business profile
1. The Business Profile  Analysis(38)  1.2844884
2. Sources of Information for the Business Profile(45)  1.2844884
3. Business Diversions(74)  0.28824288
4. Diverting Business to Vendors(3)  0.2305943
5. WHAT IS THE FINANCIAL CONDITION OF THE BUSINESS?(43)  0.2305943
6. BUSINESS REPORTING COMPANIES(49)  0.2305943
7. HOW IS THE BUSINESS ORGANIZED, LEGALLY AND STRUCTURALLY?(39)  0.20177001
8. PRINCIPALS, EMPLOYEES, AND RECORDS OF SUSPECT BUSINESS(46)  0.20177001

Doc results for: Preparing the vertical analysis
1. The Business Profile  Analysis(38)  0.41790554

Doc results for: Interviewing of the co-conspirator
** no docs returned **

conceptDocs: {49=1, 3=1, 38=2, 39=1, 74=1, 43=1, 45=1, 46=1}
Stem (lower case): in proving corrupt payments, the fraud examination often begins with

Doc results for stem: in proving corrupt payments, the fraud examination often begins with
1. Proving OnBook Payments(50)  0.349766
2. The Corrupt Recipient(31)  0.17838113
3. Methods of Proving Corrupt Payments(37)  0.17700312
4. EXAMINATION FROM THE POINT OF RECEIPT(63)  0.17128104
5. ACCOUNTING BOOKS AND RECORDS(54)  0.14522481
6. The Business Profile  Analysis(38)  0.13609743
7. Loans(25)  0.11591018
8. The Corrupt Payer(32)  0.101687975
9. Bribery(1)  0.1012375
10. Proving Payments in Cash(62)  0.10036731

option selected: 2
Incorrect.  Correct answer: Preparing the business profile
```

Figure 4.10: Concept Match V1: An Example Where A Document is Returned/Ranked for More than One Option

rank score.[10] Explaining this a bit more rigorously, for each concept document, D, with score, S, with respect to option O, if there is already an entry in the map for which the there is key/value pair, $D = (O', S')$, where O' is a different question option and S' is the score for D with respect to O', then scores, S and S' are compared, such that the option whose score is greater is chosen for D. If $S' > S$, then

[10]The Lucene scoring algorithm is normalized so that scores for documents from different queries may be compared.

```
Bribery and Corruption 17

Bribery and Corruption

in proving corrupt payments, the fraud examination often begins with which of the following?

a) Interviewing the target
b) Preparing the business profile
c) Preparing the vertical analysis
d) Interviewing of the co-conspirator

Doc results for: Interviewing the target
** no docs returned **

Doc results for: Preparing the business profile
1. The Business Profile  Analysis(38)  1.2844884
2. Sources of Information for the Business Profile(45)  1.2844884
3. Business Diversions(74)  0.28824288
4. Diverting Business to Vendors(3)  0.2305943
5. WHAT IS THE FINANCIAL CONDITION OF THE BUSINESS?(43)  0.2305943
6. BUSINESS REPORTING COMPANIES(49)  0.2305943
7. HOW IS THE BUSINESS ORGANIZED, LEGALLY AND STRUCTURALLY?(39)  0.20177001
8. PRINCIPALS, EMPLOYEES, AND RECORDS OF SUSPECT BUSINESS(46)  0.20177001

Doc results for: Preparing the vertical analysis
1. The Business Profile  Analysis(38)  0.41790554

Doc results for: Interviewing of the co-conspirator
** no docs returned **

optionsDocs: {49=1, 3=1, 38=1, 39=1, 74=1, 43=1, 45=1, 46=1}

Stem (lower case): in proving corrupt payments, the fraud examination often begins with

Doc results for stem: in proving corrupt payments, the fraud examination often begins with
1. Proving OnBook Payments(50)  0.349766
2. The Corrupt Recipient(31)  0.17838113
3. Methods of Proving Corrupt Payments(37)  0.17700312
4. EXAMINATION FROM THE POINT OF RECEIPT(63)  0.17128104
5. ACCOUNTING BOOKS AND RECORDS(54)  0.14522481
6. The Business Profile  Analysis(38)  0.13609743
7. Loans(25)  0.11591018
8. The Corrupt Payer(32)  0.101687975
9. Bribery(1)  0.1012375
10. Proving Payments in Cash(62)  0.10036731

Options doc with best match: The Business Profile  Analysis(38)
Option selected: b) Preparing the business profile
Correct!
```

Figure 4.11: Addressing the Problem of Multiple Options for a Document

the entry, $D = (O', S')$, is left as is in the map; otherwise, it is overwritten with $D = (O, S)$.

For the "Bribery and Corruption" example discussed below, we see in Fig. 4.11 that Concept Match V2 properly associates the document, "The Business Profile Analysis" (id = 38), with the correct option, "b) Preparing the business profile". The printout of the conceptDocs hash map shows the correct key/value association, 38=1, signifying that document 38 is now associated with option b, as opposed to option c.

```
Total questions answered: 150
Number question correctly answered: 101(0.6733333333333333)
number of questions where option selected is -1 (no selection): 3
```

Figure 4.12: Performance of Concept Match V2 on Definition Questions

4.4.2.1 Concept Match V2 Performance

Next, we look at the performance of the Concept Match Version 2 algorithm on the same definition questions that were used to test Concept Match V1. Fig. 4.12 displays output from the algorithm tester showing Concept Match V2 correctly answers 101 questions of the of the 150 definition questions, resulting in a score of 67.3% – a dramatic improvement over Concept Match V1. We also see that this algorithm has a much lower population of unanswered questions, (3 instead of 46), as a result of the fallback query measure incorporated into V2.

Fig. 4.13 shows we can reject the null hypothesis, H0, in favor of the alternative Hypothesis, Ha, that Concept Match V2 offers improved performance over Version 1's Max Frequency algorithm, whose accuracy is 48%[11] on the same 150 questions, at the 99% significance level.

4.4.3 Concept Match Version 3

Concept Match V3 attempts to build on Concept Match V2 by leveraging a behavior that was noticed in the results of the Lucene search results of the stem query. During development, it was noticed in a plurality of cases that return sets for the question-stem query were headlined by a document whose score was head-and-shoulders above the rest in the return set, sometimes by a factor of three or more. In these cases, it was common this document was, in fact, the correct one, containing the answer to the question. So, when we have such a document, hereafter referred

[11]see Fig. 4.27 to view the accuracy of Max Frequency on the reclassified training set of 150 definition questions.

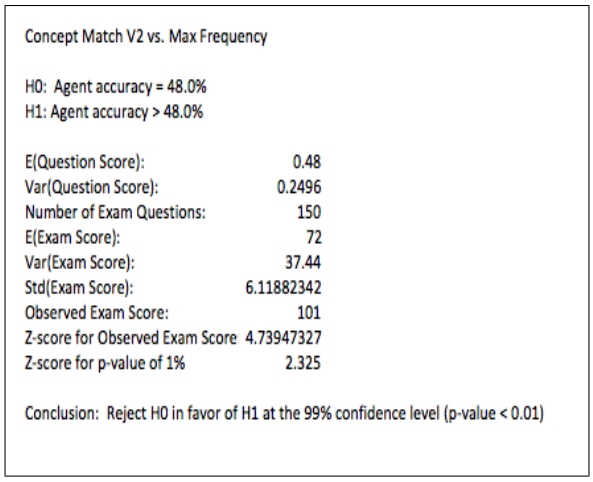


Figure 4.13: Concept Match V2 vs. Max Frequency Hypothesis Test on Definition Questions

to as a "premier document", the algorithm should focus on finding the option most closely associated with that premier document. Concept Match V3 implements this approach in questions who are found to have a premier document by choosing the option for which the premier document scores highest. If no option queries based on the title field yield the premier document, then the algorithm repeats the option queries against the contents field. If the premier document *still* does not appear in any result set, the algorithm returns -1, representing no selection.

Consider the example in Fig. 4.14 and Fig. 4.15. The question stem query for this "Criminal Prosecutions for Fraud" question returns a result set in which the "Arraignment" document, (id=12), with a score of 0.4654 outscores the next place document, "Sentencing" (id=45) by a facor of more then 2.5x. The algorithm, therefore, categorizes this document as a premier document, and thus approaches the option selection process by attempting to find the option whose query result ranks this document higher than that for any other option. In this example, we see that for the option queries based on the title field, the result sets are thin and there's no match to the premier document. So, the algorithm re-issues the option queries,

```
Question 22 of 26:
Criminal Prosecutions for Fraud 39

Criminal Prosecutions for Fraud

In certain circumstances, the defendant may be allowed to plead guilty, although continuing to assert his or her innocence. This procedure is called:

a) The Brady plea
b) The Alford plea
c) The Johnson plea
d) The Katz plea

Stem (lower case): in certain circumstances, the defendant may be allowed to plead guilty, although continuing to assert his or her innocence. this procedure

Doc results for stem: in certain circumstances, the defendant may be allowed to plead guilty, although continuing to assert his or her innocence. this procedure
1. Arraignment(12)  0.46549812
2. Sentencing(45)  0.15588728
3. DUPLICITY(25)  0.108737275
4. The Trial Process(33)  0.06501623
5. The Charging Process(8)  0.064099945
6. A MOTION CHALLENGING THE SUFFICIENCY OF THE INDICTMENT(24)  0.06306724
7. The Burden of Proof in Criminal Trials(15)  0.053804673
8. Arrest and Interrogation(6)  0.047518227
9. Prosecutorial Discretion and Plea Bargains(13)  0.04420231
10. Appeal(52)  0.040197868

Doc results for: The Brady plea
1. Prosecutorial Discretion and Plea Bargains(13)  0.7560996
2. Exculpatory Information (Brady Material)(31)  0.7560996

Doc results for: The Alford plea
1. Prosecutorial Discretion and Plea Bargains(13)  0.69746906

Doc results for: The Johnson plea
1. Prosecutorial Discretion and Plea Bargains(13)  0.69746906

Doc results for: The Katz plea
1. Prosecutorial Discretion and Plea Bargains(13)  0.69746906

Stem doc is premier: Arraignment(doc=12 score=0.46549812)
Searching for option whose matching doc is the first stem doc.
Search for options docs based on title field unsuccessful.
Repeating search for options docs using contents field...
```

Figure 4.14: Concept Match V3 Example - Part 1

this time against the contents field of each document in the collection. In Fig. 4.15, we see that this approach prevails – the result set for the correct answer, option b, the Alford plea, includes the Arraignment document. Further, we see that although this document appears in the result sets for other options' queries, it scores highest for the Alford plea option.

Fig. 4.16 shows the "Arraignment" document. It provides a couple of key insights as to the behavior of the algorithm on this particular question. First, we notice that the answer to the question is provided on lines 39 through 42. The brief segment shown here concerning a description of the Alford plea suggests (and, in

```
Doc results for: The Brady plea
1. Exculpatory Information (Brady Material)(31)  0.3185585
2. Arraignment(12)  0.24279846
3. Prosecutorial Discretion and Plea Bargains(13)  0.13765378
4. Disclosures by the Defendant(32)  0.12978123
5. Motion to Suppress Evidence(18)  0.09733592

Doc results for: The Alford plea
1. Arraignment(12)  0.7828563
2. Prosecutorial Discretion and Plea Bargains(13)  0.13765378
3. Disclosures by the Defendant(32)  0.12978123
4. Motion to Suppress Evidence(18)  0.09733592

Doc results for: The Johnson plea
1. Arraignment(12)  0.22012393
2. Prosecutorial Discretion and Plea Bargains(13)  0.12479854
3. Disclosures by the Defendant(32)  0.117661186
4. Motion to Suppress Evidence(18)  0.08824589

Doc results for: The Katz plea
1. Arraignment(12)  0.22012393
2. Prosecutorial Discretion and Plea Bargains(13)  0.12479854
3. Disclosures by the Defendant(32)  0.117661186
4. Motion to Suppress Evidence(18)  0.08824589

Search successful for option whose matching doc is first stem doc: b) The Alford plea
Option selected: b) The Alford plea
Correct!
```

Figure 4.15: Concept Match V3 Example - Part 2

fact, it turns out to be the case upon further checking) that there is no document in the collection specifically dedicated to (and whose title field would be) the Alford plea. Second, a cursory examination of this document reveals that there are number of occurrences of the term, "plea" throughout the document which would explain why this document appears in the result set for not only the Alford plea option, but also for the Brady plea, Johnson plea, and Katz plea options. However, the name, "Alford" is the only one of these names mentioned in this document, and this explains why for the Alford plea option, this document scores significantly higher than for any other option.

18
19 Arraignment
20 Once the defendant is formally charged, he is brought before the court to enter a plea. This
21 process is called the arraignment. A defendant named in an indictment, if not already in
22 custody, may be arrested on a warrant. Alternativelymore often in white-collar crime
23 cases the defendant is summoned to appear before a magistrate at a stated time and place
24 to be arraigned.
25
26 The arraignment must take place in open court, and it consists of reading the indictment or
27 information to the defendant and calling on him to enter a plea. The defendant may plead
28 guilty, not guilty, or nolo contendere. If the defendant pleads guilty, the sentencing phase of the
29 criminal justice process begins. A plea of not guilty sets in motion the adjudicative process as
30 described below. A plea of nolo contendere means the defendant does not contest the charges,
31 without formally admitting or denying them. A defendant may plead nolo only with the
32 consent of the court. If accepted, a nolo plea is the same as a plea of guilty for purposes of
33 punishment, but it cannot be used as a formal admission of guilt. This makes it a favored
34 plea for corporate defendants facing subsequent civil litigation.
35
36 Before the court will accept a guilty plea, it must follow procedures to ensure that the plea is
37 voluntary and accurate; that is, that there is a factual basis for the plea. This usually means
38 that the defendant must admit to committing acts that satisfy each element of the offense. In
39 some circumstances, however, a defendant may be allowed to enter an Alford plea (named
40 after the Supreme Court case that upheld the practice) under which he pleads guilty,
41 Law Criminal Prosecutions for Fraud
42 2011 Fraud Examiners Manual 2.509
43 although continuing to assert innocence. Such a plea may be made to obtain the benefits of a
44 plea agreement and to avoid potentially more dire consequences, such as the death penalty, if
45 the defendant is convicted after trial. Before the court accepts an Alford plea, it must
46 determine that there is strong evidence of guilt and that the defendant understands the
47 consequences.
48

Figure 4.16: The Arraignment Document

4.4.3.1 Concept Match V3 Performance

Fig. 4.17 shows the performance of the Concept Match V3 algorithm on the 150 definition-type questions of the training set. It shows that V3 improves on V2 slightly, getting 105 question correct compared with V2's 101. However, this improvement is not significant enough to be statistically significant at the 99% level, as the figure 4.18 shows. Nonetheless, the fact that V3 outpaces V2 means that the CFE agent will use V3 over V2 when confronted with a definition-type question, (and, as we'll see later, the agent will use this algorithm on other types

```
Total questions answered: 150
Number question correctly answered: 105(0.7)
number of questions where option selected is -1 (no selection): 4
```

Figure 4.17: Performance of Concept Match V3 on Training Set

Concept Match V3 vs. Concept Match V2

H0: Agent accuracy = 67.3%
H1: Agent accuracy > 67.3%

E(Question Score):	0.673
Var(Question Score):	0.220071
Number of Exam Questions:	150
E(Exam Score):	100.95
Var(Exam Score):	33.01065
Std(Exam Score):	5.74548954
Observed Exam Score:	105
Z-score for Observed Exam Score	0.70490077
Z-score for p-value of 1%	2.325

Conclusion: Accept H0.

Figure 4.18: Concept Match V3 vs. V2 Hypothesis Test on Definition Questions

of questions as well). And finally, lest we forget, compared with Version 1 of the agent, this algorithm extends the gains we achieved with Concept Match V2 that were, themselves, found to be statistically significant relative to Version 1 of the CFE agent.

4.4.4 Concept Match NOT

Concept Match NOT extends the logic of Concept Match V3, but turns it on its head to handle questions of the type, "Which of the following is NOT ...", where for the phrase that follows, all options are true except for one, which is the correct answer. An example of a question of this type is "Which of the following is

NOT a plea a defendant may enter at an arraignment?" [34] Concept Match NOT takes an approach that inverts the over-arching approach of the algorithms we've discussed above. Instead of looking for the option for which there's the greatest affinity between option query result sets and the question stem result set, Concept Match NOT attempts to find the option whose result set has the least affinity.

Fig. 4.19 and Fig. 4.20 show an example of this algorithm at work. First, as we saw in the agent's justification in earlier algorithms, we see the result set for the question stem query, and then those for the option queries. Then, the agent, attempts to map the overlap between the question stem result set and the option result sets by building two hash maps. The first one, the "docOptionScores" map, associates each document among the option query result sets with the option for which that document earned the highest score. The algorithm then utilizes this data to construct the second hash map, "optionScoreDocs", which consists of option/document key/value pairs whose documents are present in both the "docOptionScores" map and in the question stem result set. Using this data structure, the agent makes a selection; specifically, it selects the option whose document has the weakest affinity with the question stem result set. In this case, that's option d, skimming, since this option has no representation in the optionScoreDocs data structure, implying it has no documents that overlap with the result set of the question stem. (Looking closely, we see that whereas all of the other option queries have result sets that are non-empty, the the result set for the skimming option has no documents, and so, it's reasonable that it has the weakest affinity with the question stem.)

4.4.4.1 Concept Match NOT Performance

Fig. 4.21 shows the performance of the Concept Match NOT algorithm on Definition/NOT questions in the training set - 8 out of 27 correct, or 29.6%. It is not unreasonable to expect that this algorithm would show weaker performance than its inverted cousin, Concept Match V3, as discovering the negative, as we are attempting to do in the case for this question type, is difficult to do. In fact, we cannot reject the null hypothesis that this algorithm performs any better on Definition/NOT questions than random guessing, as shown in Fig. 4.22. Further

```
Fraudulent Disbursements 2

Fraudulent Disbursements

Which of the following is NOT a form of fraudulent disbursement?

a) Payroll schemes
b) Check tampering
c) Billing schemes
d) Skimming

Stem (lower case):  is not a form of fraudulent disbursement

Doc results for stem:  is not a form of fraudulent disbursement
1. Segregation of Duties(126)  0.40012556
2. Detection of Register Disbursement Schemes(10)  0.3340926
3. Register Disbursement Schemes(1)  0.32943964
4. Overbilling with a Nonaccomplice Vendors Invoices(74)  0.2597543
5. ASSET MISAPPROPRIATION  FRAUDULENT DISBURSEMENTS(0)  0.23733978
6. Check Tampering(15)  0.19778316
7. Periodic Review and Analysis of Payroll(127)  0.1812591
8. Forming a Shell Company(64)  0.17629585
9. Billing Schemes(62)  0.17272948
10. Check Disbursement Controls(58)  0.1727163

Doc results for: Payroll schemes
1. Detection of Payroll Schemes(116)  2.364177
2. Prevention of Payroll Schemes(125)  2.364177
3. Payroll Fraud(98)  0.94408447
4. Adding the Ghost to the Payroll(100)  0.75526756
5. Independent Payroll Distribution(117)  0.75526756
6. Periodic Review and Analysis of Payroll(127)  0.75526756
7. Indicators of Payroll Fraud(128)  0.75526756
8. Analysis of Deductions from Payroll Checks(123)  0.6608591
9. Billing Schemes(62)  0.5335261
10. PassThrough Schemes(71)  0.5335261

Doc results for: Check tampering
1. Check Tampering(15)  3.441594
2. Concealing Check Tampering Schemes(46)  2.7532752
3. Detection of Check Tampering Schemes(52)  2.7532752
4. Prevention of Check Tampering Schemes(57)  2.7532752
5. Physical Tampering Prevention(60)  0.7951444
6. Obtaining the Check(17)  0.7268665
7. Converting the Check(29)  0.7268665
8. To Whom Is the Check Made Payable?(20)  0.5814932
9. Converting the Stolen Check(36)  0.5814932
10. Check Disbursement Controls(58)  0.5814932
```

Figure 4.19: Concept Match Not Example - Part 1

investigation is required to refine this algorithm or to take a different approach with these types of questions altogether.

4.4.5 Concept Match NOTA

Concept Match NOTA leverages the logic in Concept Match V3 for concept matching, and extends it for addressing definition questions in which the last option is "none of the above". Hereafter, we'll refer to such questions as NOTA questions.

The first concern to investigate in developing this algorithm was the frequency with which the "none of the above" option was actually the correct response in

```
Doc results for: Billing schemes
1. Billing Schemes(62)  3.3741188
2. Detection of Billing Schemes(81)  2.699295
3. Prevention of Billing Schemes(89)  2.699295
4. PassThrough Schemes(71)  0.46728876
5. PayandReturn Schemes(73)  0.46728876
6. Commission Schemes(112)  0.46728876
7. Register Disbursement Schemes(1)  0.373831
8. Concealing Register Disbursement Schemes(7)  0.373831
9. Detection of Register Disbursement Schemes(10)  0.373831
10. Prevention of Register Disbursement Schemes(14)  0.373831

Doc results for: Skimming
** no docs returned **

docOptionScores: {128=option: 0 score: 0.7552675604820251, 1=option: 2 score: 0.37383100390434265,

optionScoreDocs: {0=doc=127 score=0.1812591, 1=doc=15 score=0.19778316, 2=doc=10 score=0.3340926}

Not all options present in optionScoreDocs.  Picking a missing option.

missing option selected: 3

Option selected: d) Skimming
Correct!
```

Figure 4.20: Concept Match Not Example - Part 2

NOTA questions. In order to determine this, we developed a trivial algorithm, called the None Of the Above Algorithm, which simply selects the last option, that is, the "none of the above" option, always. Then, we ran this algorithm on all 162 NOTA questions in the training set. The ratio of the correctly answered questions for this algorithm gave us our answer. Fig. 4.23 shows the performance results. We see that the "none of the above" option is under-represented as a correct answer, serving as the correct answer only 7.4% of the time. Whereas we'd expect that it should be the correct answer 25% of the time, it is, in fact, drastically under-represented as the correct answer. This served as the inspiration for the simplistic but strategic approach of this algorithm - simply remove the "none of the above option" as one of the possible options and select from the remaining three options (using the logic of Concept Match V3).

Fig. 4.24 shows an example of the execution of this algorithm. The agent

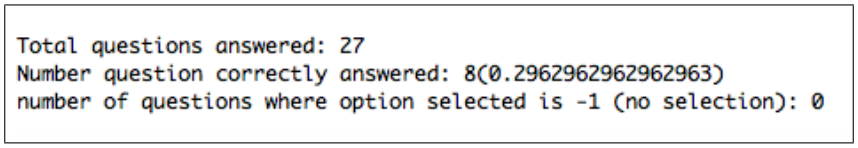


Figure 4.21: Performance of Concept Match Not on Training Set - Definition/NOT Questions

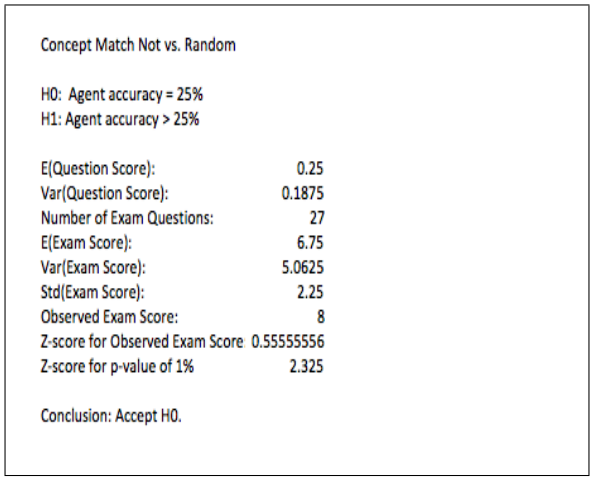


Figure 4.22: Concept Match Not vs. Random Hypothesis Test on Definition/Not Questions

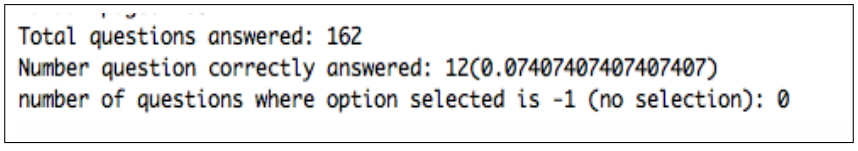


Figure 4.23: Performance of NOTA Algorithm on NOTA Questions

```
Theft of Intellectual Property 5

Theft of Intellectual Property

A special room that is acoustically shielded and radio-frequency shielded so that conversations within the room cannot be monitored outside the room is called:

a) "Quiet" room
b) "Dead" room
c) "Shield" room
d) None of the above

Stem (lower case): a special room that is acoustically shielded and radio-frequency shielded so that conversations within the room cannot be monitored outside the room

Doc results for stem: a special room that is acoustically shielded and radio-frequency shielded so that conversations within the room cannot be monitored outside the room
1. Quiet Rooms(171)  1.2052177
2. SPIKE AND CAVITY MICROPHONES(140)  0.28825757
3. Hotels(80)  0.14467405
4. Warning Signs of Bugging(144)  0.0760033
5. Travel Plans(123)  0.07587285
6. Video Surveillance(152)  0.07587285
7. Electronic Mail and Voicemail(173)  0.061087526
8. Technical Surveillance Countermeasures (TSCM) Survey(180)  0.05942014
9. ABOVE THE CEILING(149)  0.051834494
10. OTHER PLACES TO SEARCH(150)  0.03455633

This is an NONE-OF-THE-ABOVE question.  Eliminating the NONE-OF-THE-ABOVE option for this algorithm...

Doc results for: "Quiet" room
1. Quiet Rooms(171)  1.1508396

Doc results for: "Dead" room
** no docs returned **

Doc results for: "Shield" room
** no docs returned **

Stem doc is premier: Quiet Rooms(doc=171 score=1.2052177)
Searching for option whose matching doc is the first stem doc.
Search successful for option whose matching doc is first stem doc: a
Option selected: a) "Quiet" room
Correct!
```

Figure 4.24: Concept Match NOTA Example

collects the result set for the question stem query, notices that this question is a NOTA question and thus, removes the "none of the above" option from its set of options. Then, it infers from the score of the "Quiet Rooms" document (id=171) relative to that of the second place document that it's a premier document, and picks the option whose query scores the "Quiet Rooms" document higher than any other option. This leads to the correct selection of a) Quiet Rooms.

4.4.5.1 Concept Match NOTA Performance

Fig. 4.25 shows the performance of this algorithm on the Definition/NOTA questions at 65.8% accuracy. Fig. 4.26 shows a hypothesis test of this algorithm against the Max Frequency algorithm which had the best results in the CFE Agent

```
Total questions answered: 161
Number question correctly answered: 106(0.6583850931677019)
number of questions where option selected is -1 (no selection): 12
```

Figure 4.25: Performance of Concept Match NOTA on Training Set - Definition/NOTA Questions

Concept Match NOTA vs. Max Freq

H0: Agent accuracy = 56.5%
H1: Agent accuracy > 56.5%

E(Question Score):	0.565
Var(Question Score):	0.245775
Number of Exam Questions:	161
E(Exam Score):	90.965
Var(Exam Score):	39.569775
Std(Exam Score):	6.2904511
Observed Exam Score:	106
Z-score for Observed Exam Score:	2.39013065
Z-score for p-value of 1%	2.325

Conclusion: Reject H0 in favor of H1 at the 99% confidence level

Figure 4.26: Concept Match NOTA vs. Max Frequency Hypothesis Test on Definition/NOTA Questions

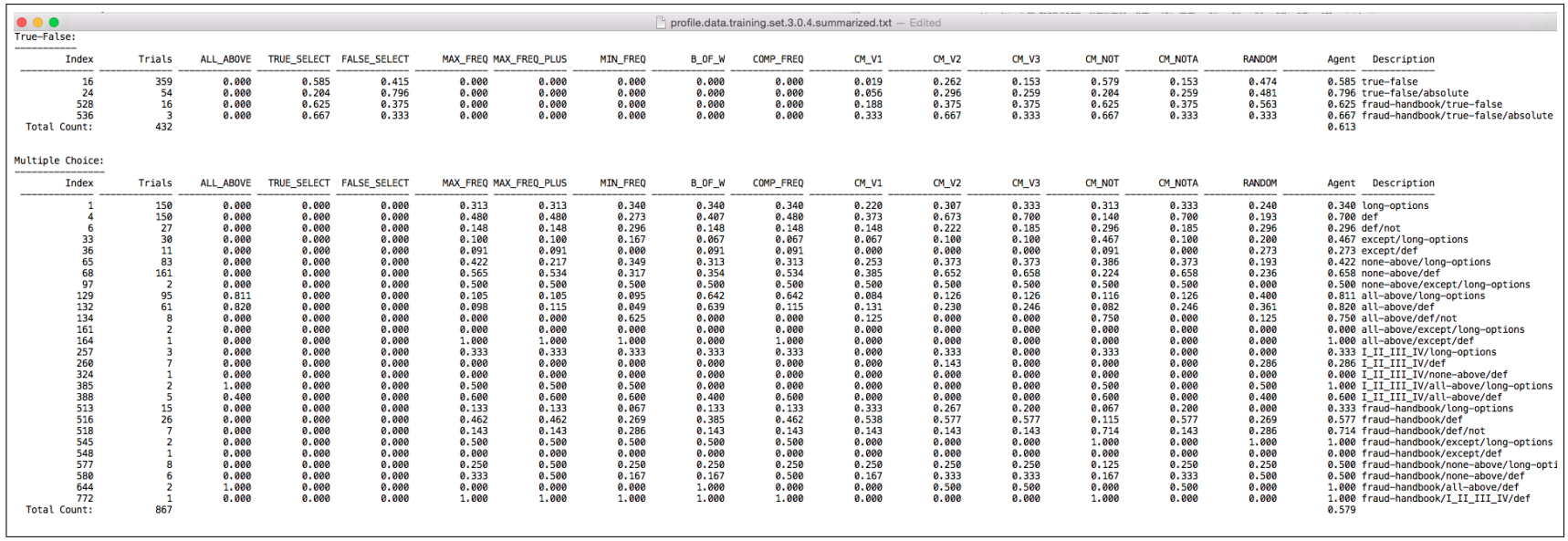
Version 1. Since Max Frequency showed results of 56.5% on 161 questions, the results of Concept Match NOTA were statistically significant at the 99% confidence level. Given this, we incorporate this algorithm into the agent as part of its enhanced suite of tools of Version 2.

4.5 CFE Agent Version 2 Results

Fig. 4.27 summarizes the performance of the CFE Agent Version 2. It shows that with a 70.0% accuracy rate on definition questions, Concept Match V3 is the preferred algorithm for that question type. Interestingly, Concept Match NOT emerges as the preferred algorithm for long-option/NOT questions, with an accuracy rate of 46.7%, even though it shows no better performance than random guessing on other question types of a similar nature – definition/NOT and definition/except. There are 30 questions in the training set of the long-option/NOT type – not an insignificant number. But further investigation is required to ascertain whether/why this particular algorithm is uniquely effective for this question type. Finally, Concept Match NOTA is the preferred algorithm for NOTA questions, with an accuracy of 65.8%.

Fig. 4.28 shows the results of a hypothesis test of the performance of the CFE Agent Version 2 on the entire battery of training set questions relative to that of Version 1. The analysis shows the improvement for Version 2 over Version 1 to be statistically significant at the 99% level. This should not be surprising since the question types for which we've targeted our new algorithms, namely Definition, Definition/NOT, and Definition/NOTA , constitute a large portion of the question battery (338 of the 867 questions), as indicated by the Trials column of the table in Fig. 4.27.

Running the CFE Agent on the test set of 200 questions yields a score of 119 out of 200, or 59.5%, a dramatic improvement over the 49% of Version 1. This shows that the more sophisticated algorithms in Version 2 offer significantly better performance, on the order of 10% marginal benefit in terms of accuracy, and offer some degree of argument-based justification, though not proof-based.



profile.data.training.set.3.0.4.summarized.txt — Edited

True-False:

Index	Trials	ALL_ABOVE	TRUE_SELECT	FALSE_SELECT	MAX_FREQ	MAX_FREQ_PLUS	MIN_FREQ	B_OF_W	COMP_FREQ	CM_V1	CM_V2	CM_V3	CM_NOT	CM_NOTA	RANDOM	Agent	Description
16	359	0.000	0.585	0.415	0.000	0.000	0.000	0.000	0.000	0.019	0.262	0.153	0.579	0.153	0.474	0.585	true-false
24	54	0.000	0.204	0.796	0.000	0.000	0.000	0.000	0.000	0.056	0.296	0.259	0.204	0.259	0.481	0.796	true-false/absolute
528	16	0.000	0.625	0.375	0.000	0.000	0.000	0.000	0.000	0.188	0.375	0.375	0.625	0.375	0.563	0.625	fraud-handbook/true-false
536	3	0.000	0.667	0.333	0.000	0.000	0.000	0.000	0.000	0.333	0.667	0.333	0.667	0.333	0.333	0.667	fraud-handbook/true-false/absolute
Total Count:	432															0.613	

Multiple Choice:

Index	Trials	ALL_ABOVE	TRUE_SELECT	FALSE_SELECT	MAX_FREQ	MAX_FREQ_PLUS	MIN_FREQ	B_OF_W	COMP_FREQ	CM_V1	CM_V2	CM_V3	CM_NOT	CM_NOTA	RANDOM	Agent	Description
1	150	0.000	0.000	0.000	0.313	0.313	0.340	0.340	0.340	0.220	0.307	0.333	0.313	0.333	0.240	0.340	long-options
4	150	0.000	0.000	0.000	0.480	0.480	0.273	0.407	0.480	0.373	0.673	0.700	0.140	0.700	0.193	0.700	def
6	27	0.000	0.000	0.000	0.148	0.148	0.296	0.148	0.148	0.148	0.222	0.185	0.296	0.185	0.296	0.296	def/not
33	30	0.000	0.000	0.000	0.100	0.100	0.167	0.067	0.067	0.067	0.100	0.100	0.467	0.100	0.200	0.467	except/long-options
36	11	0.000	0.000	0.000	0.091	0.091	0.000	0.091	0.091	0.000	0.000	0.000	0.091	0.000	0.273	0.273	except/def
65	83	0.000	0.000	0.000	0.422	0.217	0.349	0.313	0.313	0.253	0.373	0.373	0.386	0.373	0.193	0.422	none-above/long-options
68	161	0.000	0.000	0.000	0.565	0.534	0.317	0.354	0.534	0.385	0.652	0.658	0.224	0.658	0.236	0.658	none-above/def
97	2	0.000	0.000	0.000	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.000	0.500	none-above/except/long-options
129	95	0.811	0.000	0.000	0.105	0.105	0.095	0.642	0.642	0.084	0.126	0.126	0.116	0.126	0.400	0.811	all-above/long-options
132	61	0.820	0.000	0.000	0.098	0.115	0.049	0.639	0.115	0.131	0.230	0.246	0.082	0.246	0.361	0.820	all-above/def
134	8	0.000	0.000	0.000	0.000	0.000	0.625	0.000	0.000	0.125	0.000	0.000	0.750	0.000	0.125	0.750	all-above/def/not
161	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	all-above/except/long-options
164	1	0.000	0.000	0.000	1.000	1.000	1.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	all-above/except/def
257	3	0.000	0.000	0.000	0.333	0.333	0.333	0.333	0.333	0.000	0.333	0.000	0.333	0.000	0.000	0.333	I_II_III_IV/long-options
260	7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.143	0.000	0.000	0.000	0.286	0.286	I_II_III_IV/def
324	1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	I_II_III_IV/none-above/def
385	2	1.000	0.000	0.000	0.500	0.500	0.500	0.000	0.000	0.000	0.000	0.000	0.500	0.000	0.500	1.000	I_II_III_IV/all-above/long-options
388	5	0.400	0.000	0.000	0.600	0.600	0.600	0.400	0.600	0.000	0.000	0.000	0.600	0.000	0.400	0.600	I_II_III_IV/all-above/def
513	15	0.000	0.000	0.000	0.133	0.133	0.067	0.133	0.133	0.333	0.267	0.200	0.067	0.200	0.000	0.333	fraud-handbook/long-options
516	26	0.000	0.000	0.000	0.462	0.462	0.269	0.385	0.462	0.538	0.577	0.577	0.115	0.577	0.269	0.577	fraud-handbook/def
518	7	0.000	0.000	0.000	0.143	0.143	0.286	0.143	0.143	0.143	0.143	0.143	0.714	0.143	0.286	0.714	fraud-handbook/def/not
545	2	0.000	0.000	0.000	0.500	0.500	0.500	0.500	0.500	0.000	0.000	0.000	1.000	0.000	1.000	1.000	fraud-handbook/except/long-options
548	1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	fraud-handbook/except/def
577	8	0.000	0.000	0.000	0.250	0.500	0.250	0.250	0.250	0.250	0.250	0.250	0.125	0.250	0.250	0.500	fraud-handbook/none-above/long-opti
580	6	0.000	0.000	0.000	0.333	0.500	0.167	0.167	0.500	0.167	0.333	0.333	0.167	0.333	0.500	0.500	fraud-handbook/none-above/def
644	2	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.500	0.500	0.000	0.500	0.000	1.000	fraud-handbook/all-above/def
772	1	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	1.000	fraud-handbook/I_II_III_IV/def
Total Count:	867															0.579	

Figure 4.27: Performance of CFE Agent Version 2 on Training Set

CFE Agent Version 2 vs. CFE Agent Version 1
Multiple Choice Questions

H0: Agent accuracy = 47%
H1: Agent accuracy > 47%

E(Question Score):	0.497
Var(Question Score):	0.249991
Number of Exam Questions:	867
E(Exam Score):	430.899
Var(Exam Score):	216.742197
Std(Exam Score):	14.7221669
Observed Exam Score:	502
Z-score for Observed Exam Score:	4.82952005
Z-score for p-value of 1%	2.325

Conclusion: Reject H0 in favor of H1 at the 99% confidence level

Figure 4.28: CFE Agent Version 2 vs. CFE Agent Version 1 Hypothesis Test on Multiple Choice Questions

CHAPTER 5
Version 3 – A Learning Agent

In this chapter, we describe Version 3 of the CFE Agent and its development. In Version 3, the agent attempts to isolate the passage from which each question is drawn at an even finer level of detail than in Version 2. To review, Version 1 of the agent attempts to answer question using a very course-grained filter in which text is isolated only at the relatively high level of question section within the CFE Manual. Version 2 attempts to improve on this by decomposing the large sections of the manual into documents according to table-of-contents entries and specific text features, such as, capitalized sub-headings. In Version 3, the agent breaks up the text even further dividing each document into passages, and attempts to isolate the single passage most relevant to the question. It should be noted, however, that Version 3 focuses exclusively on definition-type questions, unlike Versions 1 and 2, for which performance was analyzed and discussed on many or all question types in the preceding chapters in detail. Given the narrow focus on a single passage for each question, it was thought that this approach is most appropriate for definition questions. However, extending this approach to other question types is an area for further research.

The general approach for Version 3 is the application of supervised machine learning [41], in conjunction with information retrieval for selecting the "most correct" passage for each question. The machine learning model uses logisitic regression for discriminating a correct passage from an incorrect one, based on a prescribed set of features by assigning a probability of relevance to each passage. After sorting these passages in descending order of probability, the agent selects a number of passages from the top of this list and uses a separate algorithm that selects an answer from among the four answer options. By employing machine learning, however, we must concede that justifications for answers are more opaque, as it becomes more difficult to explain the coefficients that the agent uses in the logistic regression model. Doing so would require a low level explanation connecting the questions and

their passages in the training set to the learned coefficients of the model, which is not practical. Thus the justifications for the agent's answers exclude this aspect of the algorithm.

This rest of this chapter is laid out as follows: First, we discuss the preparation of the training and test sets, and in so doing explicitly describe how each of the issues mentioned above were handled, in detail. Next, we describe the development of the logistic regression model for classifying passages as either relevant or irrelevant. Third, we discuss the application of the model to the test set and discuss its performance. And finally, we discuss two answer selection algorithms and their performance.

5.1 Development of the Training Set

This section discusses the development of the training set for the passage classification model, describing in detail the questions selected for the training set and test set, and the up-front activities that were necessary to develop the model.

5.1.1 Targeted Questions

Given the the exclusive focus on definition questions, we included only the definition questions (profile 4 in Fig. 4.27) in the training and test sets for Version 3. Fig. 4.27 indicates there are 150 such definition questions, (with a profile equal to 4), in the training set. Furthermore, it was found that among these 150 questions, 133 of them had natural language explanations from which it was easy enough to programmatically extract the page number on which the relevant passage appeared for the question using regular expressions. So, ultimately, the training set was whittled down to 133 definition questions. Using the same approach, 16 definition questions were selected from among the 200 questions in the test set. Certainly, it would have been desirable to have more questions in the test set, but it does reflect a selection percentage, 16 / 200 = 8% that is roughly similar to the selection percentage from the training set, 133 / 1300 = 10%.

5.1.2 Passage Training Set

Next, we set about the task of developing the training set of passages upon which the passage classification model is based. The approach for accomplishing this can be described as follows:

1. Determine the proper unit for breaking up the documents into passages. The choice here was to simply break up the passages along the lines of paragraphs. The rationale for this was twofold. First, paragraphs generally can be thought to contain a collection of units of meaning that make up a single larger unit, so there's a semantic rationale. Second, the practical reason – it's relatively simple to break up text by paragraph.

2. Determine the relevant passage for each question, manually. This process, of course, required simply looking through the CFE Manual to find the right paragraph. As mentioned above, using the page numbers programmatically extracted from the explanations for the questions was a huge help, here. Still, this step constituted the most tedious part of the process.

3. Use Lucene to identify the relevant documents for each question. This step utilized Lucene in the same way it was employed for Version 2. That is, elements of the stem were isolated and used as the basis for the formation of a query which was then executed against the document collection for the applicable question section. The 20 top-scoring documents were selected as the ones assumed to provide adequate coverage of the relevant passage. That is, it was presumed likely that one of the top 20 documents contained the relevant passage determined in the last step. The decision to use the number, 20, was arbitrary. However, this number proved to be sufficient in all but 8 out of the 133 training set questions.

4. Create the training set of passages. This was accomplished by extracting each passage for each of the documents from the IR step, and correctly labeling it as relevant/irrelevant. Note that for each question, the assumption was made that exactly one passage was relevant. Again, as mentioned above, for all

but 8 questions in the training set, there was one passage marked relevant. All others were marked as irrelevant. For the 133 definition questions of the training set, we generated 9419 passage records from the related documents.

5. Identify and determine the features for each passage. There were a number of features used here as inputs to the logistic regression model. Not surprisingly, we used the document search rank. The higher the rank of the containing document, the more likely it would seem that the passage would be the relevant passage for the question. Two other features were also used: the number of distinct words in common between the question stem and the passage, and the length of the longest common sequence of words between the question stem and the passage.

Note that the process of identifying the correct passage and of generating the passages was also applied to the test set. For the 16 questions in the test set, we generated 971 passage records from the related documents.

5.2 Development of the Passage Classification Model

Fig 5.1 shows the output of the program written in R [42] for constructing the passage selection classification model. As mentioned above, this model uses logistic regression to arrive at the constant coefficients shown in the figure. Some clarification of the names of the input and output variables is needed here:

1. dr – (short for "document rank") – is an input variable containing Lucene's rank of the document containing the given passage using the question stem as the query string.

2. nwic – (short for "number words in common"") is an input variable containing the number of distinct words in common between the passage and the question stem

3. llcs – (short for "length longest common sequence") is an input variable giving the length of the longest sequence of words in common between the passage and the question stem.

4. icp – (short for "is correct passage"), is the variable indicating whether the passage is the relevant passage, 1 if it is the correct passage, 0 otherwise. All of the input variables show a significant relationship with the output variable as evidened by their respective z-scores, all of which indicate significance at the 99% level of confidence.

As for the coefficients, themselves, the coefficient for the document rank variable, has a negative sign. This indicates, not surprisingly, a negative relationship with the output variable. That is, the lower the value of document rank variable, (i.e., the closer it is to the value of 1), the higher the likelihood the passage is assigned the value 1, as opposed to 0. The other two variables, nwic and llcs, both have positive coefficients, indicating a positive relationship with the output variables, as one would expect.

5.3 Application to the Test Set

The application of the classification model to the passage test set yields favorable results. Fig. 5.2 shows an extract of the data, where each record corresponds to a passage (whose text is not shown). The fields show the question id to which the passage applies, the input variables, including dr, nwic, and llcs, and the output variable, icp. On the far right is the relevance probability assigned by the model. In this case, we're not so much interested in whether this value is less than or greater than 0.5, as is commonly the case in a logistic model, but instead, its relative magnitude compared with the probabilities of the other passages for the same question. Fig. 5.2 shows the records for the top seven passages for five exam questions, ranked in order of decreasing probability for each question. The reader will notice that for each question, the passage assigned the highest relevance probability is also, in fact, the correct passage, (icp = 1), with the exception of the first question. In fact, the classification model correctly assigns the highest probability to the correct passage for 12 of the 16 questions. And it omits the correct passage from its top seven passages (this number determined as reasonable from observation of the training set) for only one out of the collection of 16 test questions, as evidenced in Fig. 5.3. In other words, its recall is 93% using a capture size of seven.

```
> # build logistic regression model for passages based on training set.
> glm.fit = glm(icp~dr+nwic+llcs,data=passages.train,family=binomial)

> summary(glm.fit)

Call:
glm(formula = icp ~ dr + nwic + llcs, family = binomial, data = passages.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6811  -0.1076  -0.0421  -0.0136   4.7312

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.10782    0.29280 -14.029  < 2e-16 ***
dr          -0.40287    0.05425  -7.426 1.12e-13 ***
nwic         0.22068    0.04908   4.497 6.91e-06 ***
llcs         0.48648    0.07650   6.359 2.03e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1328.88  on 9418  degrees of freedom
Residual deviance:  713.87  on 9415  degrees of freedom
AIC: 721.87

Number of Fisher Scoring iterations: 10
```

Figure 5.1: Passage Classification Model Summary

5.4 Answer Processing Algorithms for Version 3

The answer processing algorithms discussed in this section leverage the top seven passages extracted using the passage classification logistic regression model. Both of these algorithms are relatively straight-forward in nature and utilize the refined text body in a significant way.

```
> print(passages.best[c("rid","qid","dr","nwic","llcs","icp","prob")])
    rid                                qid dr nwic llcs icp        prob
1     1          Bribery and Corruption 17  1    3    2   0 0.053367653
2     2          Bribery and Corruption 17  2    4    2   0 0.044877673
3     3          Bribery and Corruption 17  3    3    3   0 0.039355735
4     7          Bribery and Corruption 17  5    5    2   0 0.017194378
5     5          Bribery and Corruption 17  4    1    2   0 0.010711565
6     4          Bribery and Corruption 17  4    3    1   0 0.010243744
7     6          Bribery and Corruption 17  4    0    2   0 0.008608689
8    46                  Consumer Fraud 29  1    4    1   1 0.041427076
9    49                  Consumer Fraud 29  3    3    2   0 0.024567829
10   48                  Consumer Fraud 29  3    2    2   0 0.019799156
11   47                  Consumer Fraud 29  2    2    1   0 0.018239996
12   52                  Consumer Fraud 29  4    3    2   0 0.016556015
13   50                  Consumer Fraud 29  3    0    2   0 0.012824772
14   54                  Consumer Fraud 29  5    2    2   0 0.008943407
15   83  Contract and Procurement Fraud 14  1   16    6   1 0.874241722
16   84  Contract and Procurement Fraud 14  1    3    2   0 0.053367653
17   85  Contract and Procurement Fraud 14  2    3    2   0 0.036313572
18   86  Contract and Procurement Fraud 14  3    3    2   0 0.024567829
19   87  Contract and Procurement Fraud 14  4    2    2   0 0.013321216
20   89  Contract and Procurement Fraud 14  5    2    1   0 0.005517293
21   88  Contract and Procurement Fraud 14  4    0    1   0 0.005310114
22  115        Financial Statement Fraud 9  1    4    3   1 0.102610451
23  119        Financial Statement Fraud 9  2    0    3   0 0.030646224
24  120        Financial Statement Fraud 9  3    2    2   0 0.019799156
25  118        Financial Statement Fraud 9  2    0    2   0 0.019065963
26  117        Financial Statement Fraud 9  2    1    1   0 0.014681084
27  128        Financial Statement Fraud 9  4    2    2   0 0.013321216
28  130        Financial Statement Fraud 9  4    2    2   0 0.013321216
29  187                Money Laundering 26  1   11    9   1 0.908467037
30  189                Money Laundering 26  1    3    4   0 0.129798077
31  191                Money Laundering 26  2    4    2   0 0.044877673
32  188                Money Laundering 26  1    2    2   0 0.043256767
33  190                Money Laundering 26  1    3    1   0 0.033498366
34  195                Money Laundering 26  3    2    3   0 0.031810302
```

Figure 5.2: Application of the Passage Classification Model to the Test Set

5.4.1 MLPassage1

MLPassage1 essentially uses the same Max Joint Probability algorithm used in Version 1. That is, this algorithm calculates the geometric mean frequency of the words in each answer option within the text body for the question. Except this time, of course, the text body is much more refined to the question. We see an example of MLPassage1 in Fig. 5.4, where this algorithm successfully selects the correct answer. We also see its justification for its selection by virtue of the computations of the

```
88  714 Criminal Prosecutions for Fraud 66  1    1    2   0 0.034990656
89  713 Criminal Prosecutions for Fraud 66  1    1    1   0 0.021805697
90  715 Criminal Prosecutions for Fraud 66  1    1    1   0 0.021805697
91  720 Criminal Prosecutions for Fraud 66  2    0    2   0 0.019065963
92  783 Criminal Prosecutions for Fraud 73  2    5    4   0 0.134206944
93  784 Criminal Prosecutions for Fraud 73  2    4    3   0 0.071000672
94  780 Criminal Prosecutions for Fraud 73  1    4    2   0 0.065679627
95  781 Criminal Prosecutions for Fraud 73  1    3    2   0 0.053367653
96  782 Criminal Prosecutions for Fraud 73  1    2    2   0 0.043256767
97  786 Criminal Prosecutions for Fraud 73  4    3    3   0 0.026653209
98  785 Criminal Prosecutions for Fraud 73  3    3    2   1 0.024567829
99  847       Legal Rights of Employees 23  1    9    5   1 0.476963148
100 857       Legal Rights of Employees 23  3    4    4   0 0.076717407
101 854       Legal Rights of Employees 23  2    4    2   0 0.044877673
102 849       Legal Rights of Employees 23  2    3    2   0 0.036313572
103 853       Legal Rights of Employees 23  2    2    2   0 0.029333585
104 851       Legal Rights of Employees 23  2    2    1   0 0.018239996
105 852       Legal Rights of Employees 23  2    2    1   0 0.018239996
106 914       Legal Rights of Employees 43  1    5    5   0 0.273902084
107 916       Legal Rights of Employees 43  2    4    2   0 0.044877673
108 922       Legal Rights of Employees 43  4    4    3   0 0.033017091
109 918       Legal Rights of Employees 43  2    2    2   0 0.029333585
110 915       Legal Rights of Employees 43  1    2    1   0 0.027044315
111 917       Legal Rights of Employees 43  2    2    1   0 0.018239996
112 919       Legal Rights of Employees 43  3    3    1   1 0.015248275

> # print out the number of records for which we have icp (is correct passage) = 1.
> # in this case, if we have all of them, the number should be 16. .... [TRUNCATED]
[1] 15
```

Figure 5.3: Test Set - Count of Correct Passages Retrieved By Classification Model

geometric mean and its selection of the option with the highest value. However, we are also reminded in this justification that the text body on which it is performing these calculation results from a logistic regression model whose justification is, at best, opaque.

Figure 5.4: MLPassage1 Algorithm - Test Case 1

5.4.1.1 MLPassage1 Performance

For the test set of 16 questions, this algorithm answered 8 questions correctly, 50%. Despite the refined approach to selecting passages, this algorithm did not perform up to expectations. Reasons for this are discussed below.

For a number of other questions in the test set, the algorithm failed to select the correct answer, as shown in the Fig. 5.5. As shown in the answer justification, the problem here stems from the fact that because there is no smoothing (such as, Laplace smoothing) of word frequencies, those options with at least one word that

does not occur in the narrowly focused text body are assigned a score of zero. In the case of the example shown here, the word, "scam" does not appear anywhere in the text body, and thus all options are assigned a score of zero since all of them include the word "scam". Other problems are made apparent in the test set, as well. MLPassage1 also includes no stemming of the words in order to account for word transformations between the options and the text body.

Finally, we must remember that only one of the passages is actually considered the correct passage among the seven passages extracted by the passage classification model. The other six are not relevant. To the extent MLPassage1 includes all of the passages in its text body for each question, these other six may have a deleterious effect on the performance of the algorithm. The next algorithm takes this phenomenon into account by using only the first passage among the seven (that is, the passage with highest probability of relevance).

5.4.2 MLPassage2

MLPassage2 addresses two of the problems cited with MLPassage1, namely the stemming problem and the passage selection problem. An implementation of the Porter Stemmer algorithm was incorporated into this algorithm to address the first issue. As for the second, this algorithm includes only the first text passage out of the seven. As noted above, the passage classification algorithm has good precision. So, reducing the text body to only the first passage would appear to be a prudent decision. The results bear this out – With these modifications, the MLPassage2 algorithm shows improved performance of 11 of 16 correct (68.75%), as shown in Fig. 5.6.

The performance of MLPassage2 is comparable to that seen in Version 2 of the agent on definition questions. In other words, we see, here, that more complexity doesn't necessarily translate to better accuracy. After all, we've incorporated another step, using machine learning, which performs its function quite well. Yet, that alone is not enough to translate to a better score on the exam questions. There is more room for enhancements and modifications to MLPassage2, however, which should result in better performance with further research.

```
Question 2 of 16:
Consumer Fraud 29

Consumer Fraud

"Boiler rooms," "fronters," "closers," and "verifiers" are all terms associated with which of the following?

a) Real estate scams
b) Telemarketing scams
c) Advance fee frauds
d) Internet fraud

executing mlpassage algorithm for Consumer Fraud 29...

Analysis of option: [real, estate, scams]
real: 1
estate: 1
scams: 0
Total Frequency for Option: 0.0
Geom Mean Frequency: 0.0

Analysis of option: [telemarketing, scams]
telemarketing: 3
scams: 0
Total Frequency for Option: 0.0
Geom Mean Frequency: 0.0

Analysis of option: [advance, fee, frauds]
advance: 0
fee: 0
frauds: 0
Total Frequency for Option: 0.0
Geom Mean Frequency: 0.0

Analysis of option: [internet, fraud]
internet: 0
fraud: 4
Total Frequency for Option: 0.0
Geom Mean Frequency: 0.0
1: Consumer Fraud 29 1 4 1 1 0.0414270763706494 boiler room staff work in a boiler room is shared by fronters, closer
2: Consumer Fraud 29 3 3 2 0 0.0245678290637993 the salespeople in boiler rooms are sometimes as desperate as their v
3: Consumer Fraud 29 3 2 2 0 0.0197991561758464 staff exploitation the customers of fraudulent telemarketing operatio
4: Consumer Fraud 29 2 2 1 0 0.0182399960989958 closers the closer is a veteran. fronters pass an interested caller t
5: Consumer Fraud 29 4 3 2 0 0.0165560148868311 900 numbers/800 numbers/international calls 900 numbers are usually a
6: Consumer Fraud 29 3 0 2 0 0.0128247723027334 naturally, there are no social security or payroll taxes deducted fro
7: Consumer Fraud 29 5 2 2 0 0.00894340731137933 other common hustles occur in the privacy of ones own home or throug
Option selected: a) Real estate scams
Incorrect.  Correct answer: Telemarketing scams

Explanation: Terms in the telemarketer??s vocabulary include banging, or nailing, the customer (i.e., closing the dea
Manual page: 1.1709
```

Figure 5.5: MLPassage1 Algorithm - Test Case 2

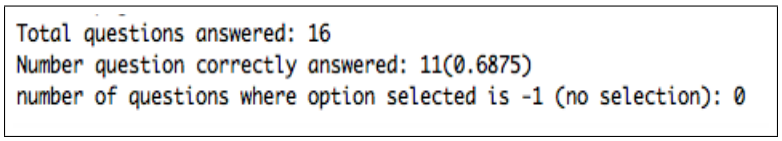


Figure 5.6: MLPassage2 Algorithm Performance

CHAPTER 6
Version 4 – Toward a Formal Model for Fraud Detection

In this chapter, we attempt to lay the groundwork for the rigorous study of fraud detection. Our goal is to not to build an agent in the same sense as Versions 1, 2, and 3 of the CFE Agent whose performance is to be evaluated against CFE exam questions in the spirit of psychometric AI, but instead to build a demonstration of a formalized approach to the domain of fraud detection, whereby through formal reasoning, we can answer questions with logico-deductive proof-based justifications. Furthermore, our ambition in this chapter is not to create a *comprehensive* formal model, but to demonstrate the technique on an examplar subdomain - the detection of doctor shopping. Our ultimate aim is to show the behavior of an automated agent if it were built under this approach. By doing so, we give an idea of the costs levied in terms of complexity[12] in order to achieve proof-based justifications.

Included in this chapter is an analysis of the doctor shopping fraud activity in which we present a computational model of doctor shopping in terms of a formal description of the behavior of the perpetrator and the doctors he targets in perpetrating this fraud activity using the $\mathcal{DCEC}^*$ [43, 44]. The $\mathcal{DCEC}^*$ provides a framework for modeling the interactions of agents in multi-agent systems with respect to their knowledge, beliefs, perceptions, plans, and natural-language capacity. The syntax and inference rules of the $\mathcal{DCEC}^*$ are described in detail in [43, 44]; this machinery will be used heavily here. We specifically use the version of the $\mathcal{DCEC}^*$ described in [44].

The modeling is done in the Slate [45] graphical proof-construction environment, where proof verification and proof discovery can be accomplished in an easy-to-use modality. Slate is based on natural deduction, and includes support for constructing proofs in propositional logic, first-order logic ($\mathcal{FOL}$), and several modal

Portions of this chapter previously appeared as: J. Johnson, N. S. Govindarajulu, and S. Bringsjord, "A three-pronged Simonesque approach to modeling and simulation in deviant bi-pay auctions, and beyond," *Mind & Society*, vol. 13, no. 1, pp. 59–82, June 2014

[12]Lacking an agent whose performance can be evaluated on exam questions, we cannot measure the cost in terms of accuracy.

logics. Slate also has the ability to automatically discover proofs via resolution, by calling ATPs; for example, SNARK [46]. This feature allows one to utilize Slate in a hybrid mode to construct proofs that are semi-automated. Slate is ultimately based on a mathematical model of computation and reasoning that is a generalization and extension of Kolmogorov-Uspenskii machines [47, 48].

6.1 An Example Subdomain - Doctor Shopping

Doctor shopping is the illegal activity of procurring multiple prescriptions from multiple doctors for the purpose of abuse or illicit profit. In some cases, the doctors are unaware that the patient has procurred the same prescriptions being sought from other doctors. In other cases, doctors are co-conspirators. In this example, we focus on the case in which the patient is the sole perpetrator and the doctors are unaware of his illicit activity. The CFE Manual describes doctor shopping briefly as follows:

> Doctor/ER Shopping: Excessive drug claims for controlled substance drugs. Patient shops for controlled substance drugs. One physician does not know that the other has prescribed the drug. In addition, the patient may shop for drugs in emergency rooms complaining of soft tissue injuries, sprains, and strains.[49]

In the development of our formal model for this subdomain, our perspective is that of a fraud examiner; that is, an agent whose purpose is to detect doctor shopping, and to provide justification for its reasoning. Formal reasoning models provide transparent justifications in the form of applications of inference rules from axioms and declarations pertaining to the problem domain. We present axioms for the doctor shopping subdomain, and follow them with declarations for a particular example related to this subdomain, below.

Before presenting the formal axioms, though, we need to make note that any formal definition of fraud must include the notion of intent. It is not sufficient to define fraud in terms of a sequence of actions or events. For example, it's possible that an patient innocently seeks a prescription from a second doctor simply because

he decided he did not like the first doctor he saw for a particular ailment and also, lost the script the first doctor gave him, (pretending for the moment, we're in the pre-modern age of say, 10 years ago when electronic communication of prescriptions was not ubiquitous). Fraud inherently involves the intent of one party to deceive another for the purpose of personal gain. And intent implies knowledge. Thus, the definition of any fraud must inherently involve assertions about cognitive states concerning knowledge and belief on the part of the actors involved. This is where the use of the $\mathcal{DCEC}^*$ comes in handy, as we'll see in the following axioms.

6.1.1 Doctor Shopping Axioms

The first axiom poses a definition of doctor shopping as follows: Any person, x, is guilty of doctor shopping if and only if there exist two doctors, $d1$ and $d2$, such that x knows he gets a prescription, r, for duration, dur, from $d2$ at time $t2$ while he knows he was already given the same prescription by doctor, $d1$, at time, $t1$, where $t1$ is prior to $t2$ and where the time between $t1$ and $t2$ is short relative the duration of the prescriptions. (In this case, we arbitrarily specify "short" as less than 10% of the prescription duration). The spirit of this definition is to capture those activities in which the perpetrator is extracting relatively large prescriptions in short order from multiple doctors. It does not, however, attempt to cover, say, addicts who shop from doctor to doctor looking for prescriptions of short duration simply to support their addiction - (this sort of behavior is emphalso sometimes referred to as doctor shopping.) A formal expression of the first axiom is as follows:

$$
\begin{aligned}
[A1]\ \forall x.\ (&guilty(x, Doctorshopping)) \Leftrightarrow \\
&\exists r, d1, d2, dur, t1, t2.\ (\mathbf{K}(x, happens(action(x, getrx(r, d1, dur)), t1)) \\
&\wedge \mathbf{K}(x, happens(action(x, getrx(r, d2, dur)), t2)) \\
&\wedge \mathbf{B}(x, \neg\mathbf{K}(d2, happens(action(x, getrx(r, d1, dur)), t1))) \\
&\wedge prior(t1, t2) \\
&\wedge duration(t1, t2) < dur \times 0.10
\end{aligned}
$$

Our first axiom provides a narrow but fine-grained definition of doctor shopping not only in terms of action but also in terms of the cognitive states of our perpetrator and his unwitting accomplices. However, for our fraud examiner agent, detecting mental states can be problematic as the data presented, whether it be from a medical claims database or a scenario described in natural language text, often does not provide that level of detailed information. This leaves our agent having to make some assumptions about the mental state of our perpetrator. So, our agent makes a general assumption of sound mind – that is, if an actor commits an act, she is aware, or knows, she is committing that act. Hence, we have the second axiom given below, that states that if a person, x, gets a prescription of of duration, r, from doctor, d, at time t, he knows he's getting that prescription.

$$[A2]\ \forall x, r, d, dur, t.\ (happens(action(x, getrx(r, d, dur)), t) \Rightarrow \mathbf{K}(x, (happens(action(x, getrx(r, d, dur)), t))))$$

In keeping with the general assumption of sound mind, we pose the third axiom, which states that if a patient does not tell a second doctor, $d2$, at time $t2$, that he already received a prescription, r, from doctor, $d1$, at time, $t1$, where $t1$ is prior to $t2$, then he knows does not do so. To emphasize, we're making the assumption, here, that such behavior doesn't arise by accident due to sheer absent-mindedness, but instead due to a deliberate attempt to deceive.

$$[A3]\ \forall x, r, d1, d2, dur, t1, t2.\ \neg happens(action(x, tells(d2, happens(action(x, getrx(r, d1, dur)), t1))), t2) \wedge prior(t1, t2) \wedge d1 \neq d2 \Rightarrow \mathbf{K}(x, \neg happens(action(x, tells(d2, happens(action(x, getrx(r, d1, dur)), t1))), t2))$$

6.1.2 Doctor Shopping - An Example

Suppose we have the following scenario:

> Blue gets a prescription from Dr. White on Wednesday for 90 day supply of escitalopram (lexapro) 10mg. Blue gets a second prescription from Dr. Black on Friday of the same week for the same medication. Blue does not tell Dr. Black he received the prescription from Dr. White. Blue believes that if he does not tell Dr. Black about the prescription from Dr. White, then Dr. Black does not know about the prescription from Dr. White. Is Blue guilty of doctor shopping?

We'd like the agent to utilize its formal model for doctor shopping to determine the answer to this question. It should be noted, however, that this kind of question is one that Versions 1, 2, and 3 of the CFE Agent are not designed to handle particularly well since the stem of the question cannot be traced back to a yes or no answer to this question (with any meaningful justification) by referring to the CFE manual. A formal representation of this problem, in conjunction with the axioms for doctor shopping allow our agent to reason on a semantic level not provided for in our prior versions of the agent, however. Of course, getting from a natural language representation of the problem domain to a formal representation continues to be an outstanding problem for research, as will be discussed in Chapter 7. For now, however, we simply move forward with assuming this bridge has been crossed, by simply asserting the formal declarations that represent the semantics of the problem using the $\mathcal{DCEC}^*$.

This first declaration deals with Blue getting a 90-day prescription for Lexapro from Dr. White on Wednesday:

$$[D1]\ happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday)$$

Next, we have a similar declaration for Blue getting a 90-day prescription for Lexapro from Dr. Black on Friday:

$$[D2]\ happens(action(Blue, getrx(Lexapro, DrBlack, 90)), Friday)$$

For our third declaration, we formally represent the assertion that Blue does not tell Dr. Black on Friday that he received a 90-day prescription for Lexapro from Dr. White on the prior Wednesday.

$$[D3]\ \neg happens(action(Blue, tells(DrBlack, happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday))), Friday)$$

And finally, we have a declaration that asserts Blue believes that if he does not tell Dr. Black about the prescription from Dr. White during his visit with Dr. Black on Friday, then Dr. Black does not know Blue received the prescription from Dr. White the preceding Wednesday.

$$[D4]\ \mathbf{B}(Blue, (\neg happens(action(Blue, tells(DrBlack, happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday))), Friday) \Rightarrow \neg\mathbf{K}(DrBlack, happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday))$$

6.1.3 Doctor Shopping Proof

These declarations and the axioms for doctor shopping are sufficient for our agent to prove (and provide justification for its reasoning) that Blue is indeed guilty of doctor shopping. We walk through the proof in the following paragraphs, step by step.

First, the agent uses $[A2]$ and $[D1]$ to prove that Blue knows he was given a prescription by Dr. White on Wednesday, as shown in Fig. 6.1. Formally, this conclusion is stated as follows:

$$[C1]\ \mathbf{K}(Blue, happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday))$$

Next, the agent uses $[A2]$ and $[D2]$ to prove Blue also knows he was given a prescription by Dr. Black on Friday, as shown in Fig. 6.2, which stated formally, is as follows:

$$[C2]\ \mathbf{K}(Blue, happens(action(Blue, getrx(Lexapro, DrBlack, 90)), Friday))$$

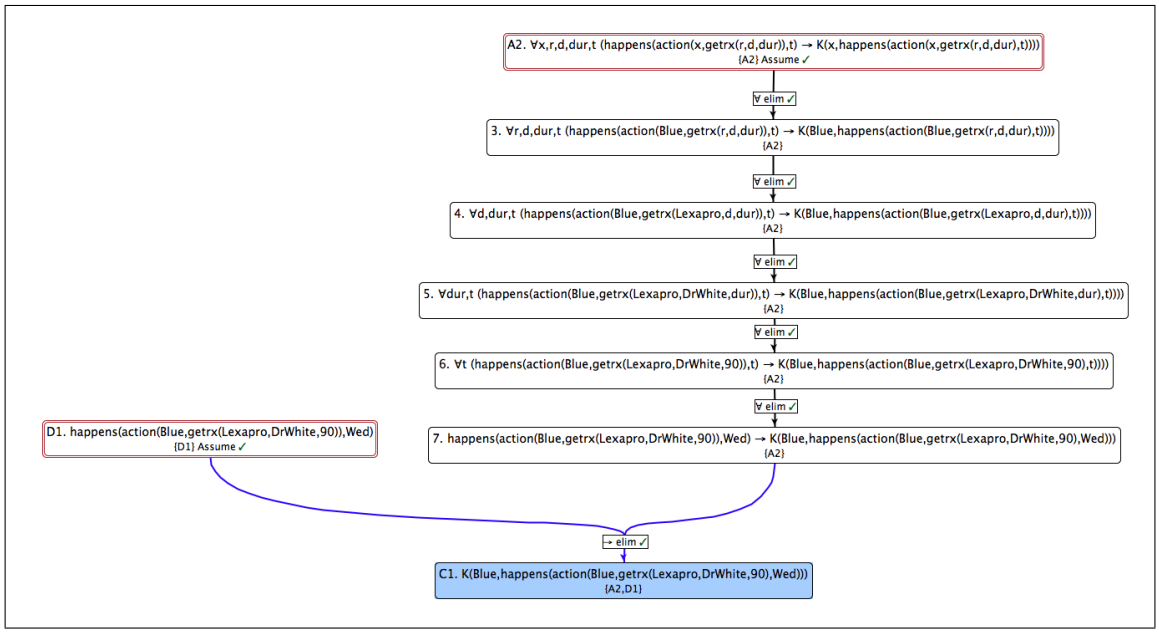


Figure 6.1: Proof of $C1$ – Blue knows he was given a prescription by Dr. White on Wednesday.

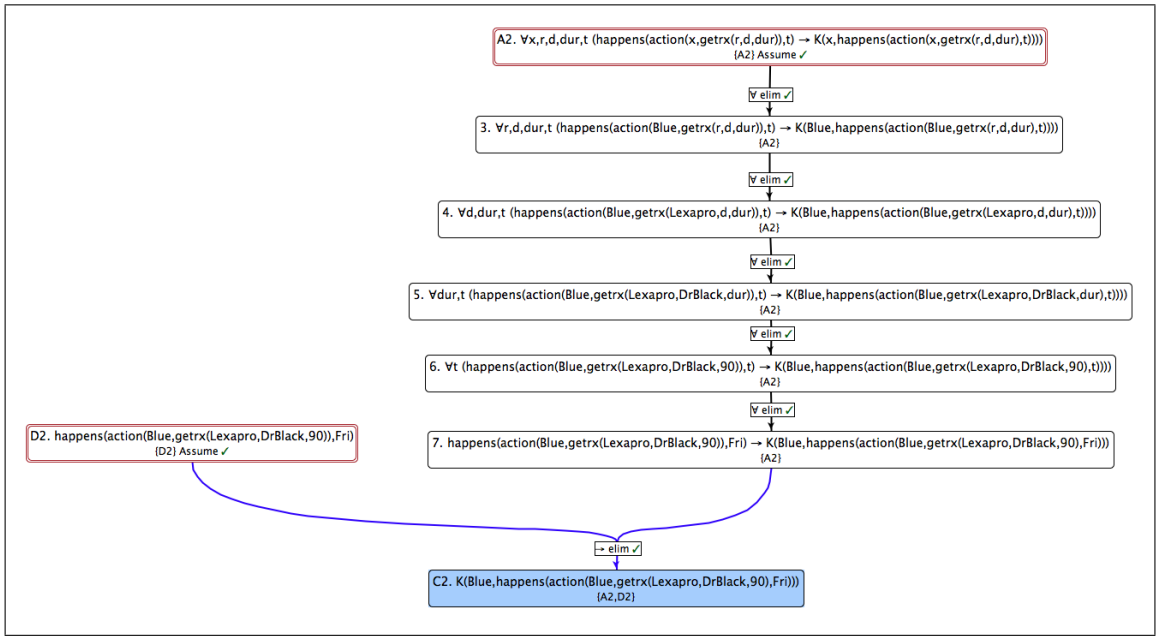


Figure 6.2: Proof of $C2$ – Blue knows he was given a prescription by Dr. Black on Friday.

Next, the agent uses [$A3$] to prove the conditional statement that if Blue does not

tell Dr. Black about the prescription he received from Dr. White, then he *knows* he does not tell him, stated formally in [C3] below. The proof is shown in Fig. 6.3.

$$
\begin{aligned}
[C3]\ \neg happens(action(Blue, tells(DrBlack,\\
&happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday))), Friday)\\
&\wedge prior(Wednesday, Friday) \wedge DrWhite \neq DrBlack \Rightarrow\\
&\mathbf{K}(Blue, \neg happens(action(Blue, tells(DrBlack,\\
&happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday))), Friday))
\end{aligned}
$$

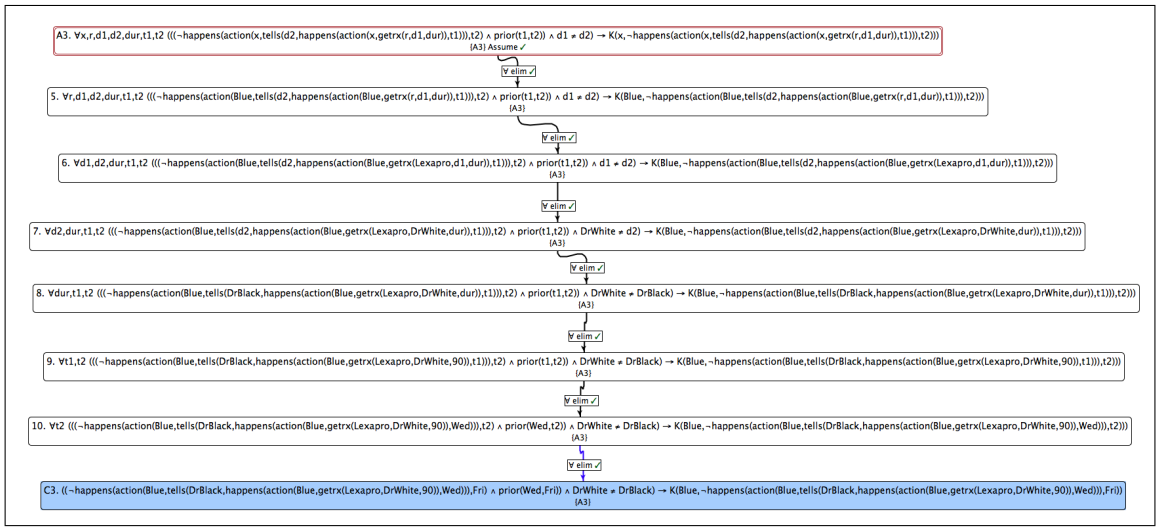


Figure 6.3: Proof of $C3$ – If Blue does not tell Dr. Black about the prescription he received from Dr. White then Blue knows he does not tell him.

Next, the agent uses $[C3]$, $[D3]$, and $[D4]$ to prove the assertion that Blue believes that Dr. Black does not know about the fact he reeived a prescription from Dr. White on Wednesday, as represented formally by $[C4]$, shown below. This proof is shown in Fig. 6.4. Also required by this proof are two supporting declations – one about the order of occurence of Wednesday and Friday in a given week and another that asserts that $DrBlack$ and $DrWhite$ are constants referencing different objects (doctors, actually) in the problem domain.

$$[C4]\ \mathbf{B}(Blue, \neg\mathbf{K}(DrBlack, happens(action(Blue, getrx(Lexapro, DrWhite, 90)), Wednesday)))$$

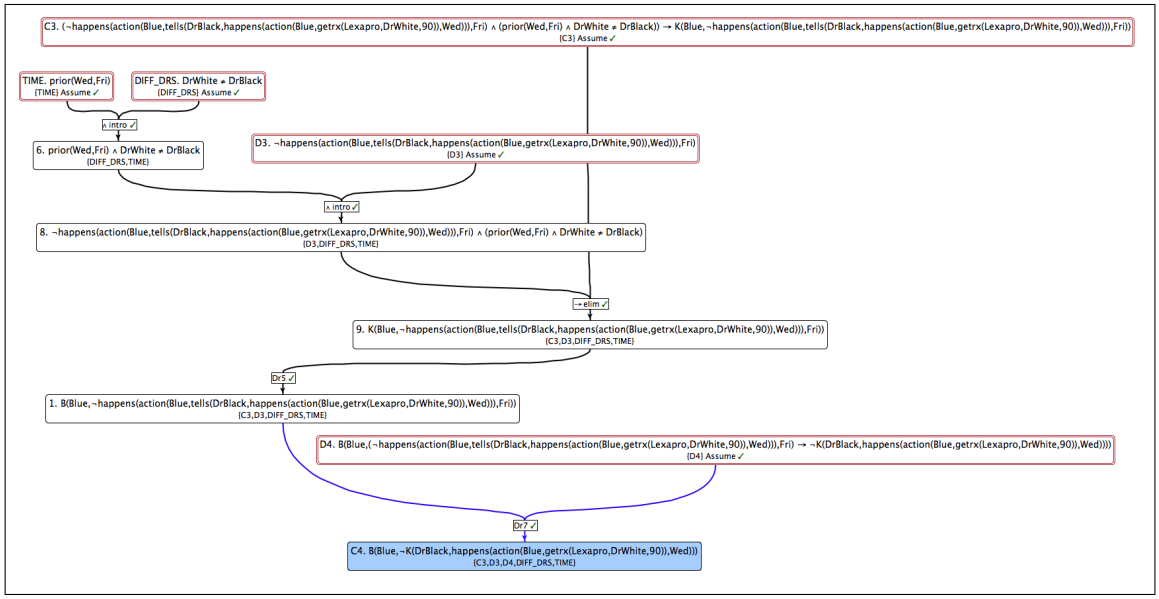


Figure 6.4: Proof of $C4$ – Blue believes that Dr. Black does not know he received a prescription from Dr. White on Wednesday.

Finally, using $[A1]$, $[C1]$, $[C2]$, and $[C4]$, the agent proves Blue is guilty of doctor shopping. As was the case for the proof of $[C4]$, the proof also requires supporting declarations related to time and the number of days between Wednesday and Friday relative to 10% of the prescription duration. This proof is shown in Fig. 6.5.

$$[C5]\ guilty(Blue, Doctorshopping)$$

Taken together, these proofs add up to a completely transparent justification of the agent's reasoning for its ultimate conclusion that Blue is guilty of doctor shopping. As mentioned earlier, this level of transparency is a feature of this formal, semantic approach that the approaches of Versions 1, 2, and 3 do not offer. However, there are costs to this approach as well, as the reader may have noted during the walk-through of the proof. First, natural language assertions must be translated

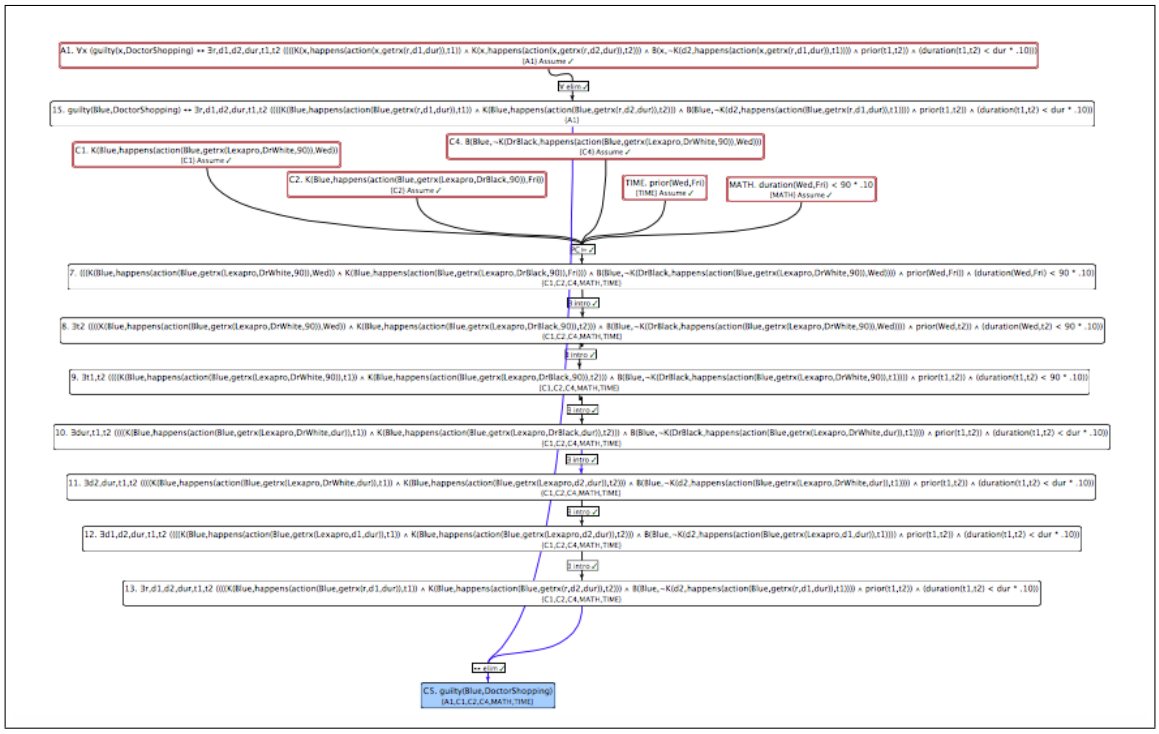


Figure 6.5: Proof of $C5$ – Blue is guilty of doctor shopping.

into a formal language, as mentioned earlier. And second, the model must be constructed such that it is comprehensive enough for the agent to make the necessary inferences it needs to complete the proof. Niether of these is an easy task. Often, assertions must be constructed based on common-sense-based real world knowledge, and are not explicitly stated in the text of supporting documentation or the scenario description. Further discussion of these issues as well as the merits and pitfalls of the approaches we've seen to this point in the development of the CFE agent will be continued in Chapter 7.

This chapter adds another data point to our comparative analysis of state-of-the-art approaches to QA in terms of the competing priorities of accuracy, complexity, and reasoning justification. In this case, the reasoning justification is higher than any of the approaches we've discussed, but comes at the cost of a great deal of complexity – the issues in the preceding paragraph should be convincing on this point.

CHAPTER 7
Conclusion and Future Work

This chapter summarizes the work weve covered in this dissertation and offers areas for further investigation.

7.1 Summary of Work

The research underpinning this dissertation began with developing a battery of multiple choice questions from the CFE exam materials and the preparation of the CFE Manual text for test processing by an automated agent. This laid the groundwork for the development of the agents discussed in detail above for which a comparative analysis of accuracy, complexity, and provision for reasoning justification could be accomplished. Version 1 of the CFE agent uses only primitive rule-based algorithms, but performs effectively, showing improved performance over random guessing at the 99% level of statistical significance. Unfortunately, justifications for the agents answers are minimal, offering little more explanation than the choice of a particular algorithm based on relative performance in the training set.

In Version 2 of the agent, more sophisticated algorithms are brought to bear on the problem using information retrieval for which we see marked improvement in accuracy (a jump from roughly 49% accuracy to 59.5%, and in particular, a 70% accuracy rate for Concept Match V3 on definition questions) and a more transparent justification for the agents answers. However, the justifications in the agent's answers are still not *completely* transparent as the coefficients upon which the cosine similarity computations are based are derived from term counts within the documents of the document collection. And so, justifications are thus limited to showing how these coefficients determined at the aggregate level are applied at the level of each individual question.

Version 3 of the agent uses machine learning to isolate the passages (paragraphs) from among the documents in the document collection relevant to each definition question. This technique, with its added complexity, provides 93% proba-

bility that the correct passage is within the selection of top seven passages as ranked by the logistic regression algorithm in the test set. Although the relatively simplistic answer extraction algorithms that utilize the results of the passage selection algorithm yield roughly 69% accuracy on the definition questions in the test set, we believe further work to refine these answer extraction algorithms will result in significant accuracy gains, surpassing those of Version 2 at a statistically significant level. However, as we have observed with Version 2, this approach offers only limited transparency into the reasoning by the agent, limited to the application of opaquely determined coefficients at the aggregate level to each individual problem.

Finally, in Version 4 of the agent, we demonstrated how an agent using formal semantic reasoning as the basis for a rigorous, mechanistic approach to fraud detection should behave, and looked at the complex issues associated with building such an agent. This approach offers the benefit of completely transparent reasoning based on natural deduction using the $\mathcal{DCEC}^*$. However, our work, here, highlights an two important issues with this approach – the problem of translating assertions expressed in natural language into a formal language, (in our case, again, the language of choice was the $\mathcal{DCEC}^*$), and that of a natural deduction theorem prover. This problem exists for both translating assertions in the information source (the CFE Manual, in our example), and for translating declarations germane to the problem at hand. In the example we covered in Chapter 6, these were the assertions associated with Blue and his interactions with Dr. White and Dr. Black.

7.2 Future Work

Certainly, as alluded to in the above paragraphs, the incorporation of machine learning into passage selection offers considerable promise for refining the algorithms of the CFE agent. So, continuing along this path, further refinement of the answer extraction algorithms would also likely deliver improved overall accuracy, as discussed above. Specifically, perhaps incorporation of machine learning in the answer extraction algorithm as well as the passage selection algorithm may prove beneficial. This is a tack that will be pursued in future work.

Another avenue for investigation is the approach for algorithm selection for a

given question. As discussed, the current approach in Version 1, 2, and 3 is to select that algorithm with the highest accuracy for the given question's type. However, there are other possibilities for selecting an algorithm/answer – one suggestion is to use a voting mechanism in which the most accurate algorithms for the question type each weigh in with a vote for a candidate answer. The answer receiving the highest number of votes weighted by the accuracy of its constituent algorithms is the one selected.

Now, consider the following problem from the CFE test set:

> True or False: Green, an agent for the White Corporation, acting within the scope of his duties and for the benefit of White, committed an act of fraud. According to the Sentencing Guidelines effective in 1991, White cannot be held criminally liable for Green's actions. [34]

This is a true/false question, a type that is not specifically targeted in any of the first three versions of the CFE agent. The reason for that is obvious this is a tough problem for the CFE Agent using the algorithms of Versions 1, 2, and 3 where the algorithms are looking for the juxtaposition of key words of the question stem and those of the various answer options within the CFE Manual. In a true/false problem, however, answer options are of no help, leaving the agent with only the question stem from which it must discern whether the statement is valid. Using proximity of key words will likely not work here as it is likely that such questions with keywords richly represented in a particular section are as likely to be invalid as valid.

An algorithm that is based on a semantic understanding of the problem would certainly appear useful, here. But for that, we'd need the tools for translating natural language assertions to ones expressed in a formal language, where in our case, we'd recommend the $\mathcal{DCEC}^*$ due to its built-in support for modeling the cognitive states and interactions typical of agents in financial fraud scenarios. There are two possible approaches for tackling this task. The first approach to the translation problem is a computational semantics approach based on Blackburn and Bos [50, 51, 52, 53, 54, 55] using Prolog [56, 57, 58, 59, 60, 61], wherein the authors explain an approach to the representation of natural language assertions in first-order

logic ($\mathcal{FOL}$) [50] using the lambda calculus [51] and inference [53, 54] from these $\mathcal{FOL}$ assertions. In the approach we propose here, however, the target representation language would be the $\mathcal{DCEC}^*$, not $\mathcal{FOL}$. A second approach would involve parsing the sentences first, using a probabilistic parser [62], such as OpenNLP [63] or the Stanford Parser [64], and then translating the parse structures into assertions expressed in the $\mathcal{DCEC}^*$. The idea here is that both the natural language utterances and the parse structures serve as inputs to the computational semantic model that produces the output assertions in the $\mathcal{DCEC}^*$ from which inferences can be drawn using a theorem prover. Having parse structures as inputs may be helpful in better capturing the semantics of the utterance in the semantic model.

As a final thought, we believe that any comprehensive QA system that provides both answers with a high degree of accuracy along with justifications, a holistic approach would be required one that leverages extended versions of the machine learning/information retrieval approaches in Versions 2 and 3 of the agent as well as the semantic approach of Version 4. How might that be accomplished? Depending on the type of question, this holistic approach could take on different forms. For true/false questions, as we have seen, machine learning approaches may be of limited use, and thus an approach more centered around semantic computation is required. However, for definition questions, a high level approach might be to leverage the machine learning approach to arrive at an answer, and then use that answer to engineer the justification using a semantic approach. The idea here is to utilize the machine learning approaches for finding the answers and then theorem proving for explaining them. For other types of questions, still yet a different blend of these approaches may be required.

REFERENCES

[1] D. R. Radev, J. Prager, and V. Samn, "Ranking suspected answers to natural language questions using predictive annotation," in *Proc. 6th Conf. Applied Natural Language Processing*. Association for Computational Linguistics, April 2000, pp. 150–157.

[2] J. Martin and D. Jurafsky, "Question answering and summarization," in *Speech and Language Processing. An introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed. Saddle River, NJ, USA: Prentice Hall, 2000, ch. 23, pp. 765–810.

[3] S. J. Russell and P. Norvig, "Introduction," in *Artificial Intelligence: A Modern Approach*, 3rd ed. Saddle River, NJ, USA: Prentice Hall, 2010, ch. 1, pp. 1–33.

[4] C. D. Manning, P. Raghavan, and H. Schütze, "Boolean retrieval," in *Introduction to Information Retrieval*, 1st ed. Cambridge, MA, USA: Cambridge Univ. Press, 2008, ch. 1, pp. 1–17.

[5] C. D. Manning, P. Raghavan, and H. Schütze, "The term vocabulary and postings list," in *Introduction to Information Retrieval*, 1st ed. Cambridge, MA, USA: Cambridge Univ. Press, 2008, ch. 2, pp. 18–44.

[6] C. D. Manning, P. Raghavan, and H. Schütze, "Index construction," in *Introduction to Information Retrieval*, 1st ed. Cambridge, MA, USA: Cambridge Univ. Press, 2008, ch. 4, pp. 61–77.

[7] C. D. Manning, P. Raghavan, and H. Schütze, "Scoring, term weighting, and the vector space model," in *Introduction to Information Retrieval*, 1st ed. Cambridge, MA, USA: Cambridge Univ. Press, 2008, ch. 6, pp. 100–123.

[8] R. Brachman and H. Levesque, "Expressing knowledge," in *Knowledge Representation and Reasoning*, 1st ed. San Francisco, CA, USA: Morgan Kauffman Publishers, 2004, ch. 3, pp. 31–48.

[9] R. Brachman and H. Levesque, "Resolution," in *Knowledge Representation and Reasoning*, 1st ed. San Francisco, CA, USA: Morgan Kauffman Publishers, 2004, ch. 4, pp. 49–84.

[10] R. Brachman and H. Levesque, "Reasoning with horn clauses," in *Knowledge Representation and Reasoning*, 1st ed. San Francisco, CA, USA: Morgan Kauffman Publishers, 2004, ch. 5, pp. 85–98.

[11] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager *et al.*, "Building Watson: An overview of the DeepQA project," *AI magazine*, vol. 31, no. 3, pp. 59–79, Fall 2010.

[12] S. Bringsjord, N. Govindarajulu, J. Licato, A. Sen, J. Johnson, A. Bringsjord, and J. Taylor, "On logicist agent-based economics," in *Proc. Artificial Economics*, 2015.

[13] J. Johnson, N. S. Govindarajulu, and S. Bringsjord, "A three-pronged Simonesque approach to modeling and simulation in deviant bi-pay auctions, and beyond," *Mind & Society*, vol. 13, no. 1, pp. 59–82, June 2014.

[14] S. J. Russell and P. Norvig, "Truth maintenance systems," in *Artificial Intelligence: A Modern Approach*, 3rd ed. Saddle River, NJ, USA: Prentice Hall, 2010, section 12.6, pp. 460–462.

[15] Association of Certified Fraud Examiners, *ACFE*. [Online] Available: http://www.acfe.com/ Accessed on: November 1, 2011.

[16] H. Markopolos, "Introduction," in *No One Would Listen*, 1st ed. Hoboken, NJ, USA: John Wiley & Sons, 2010, pp. 1–4.

[17] Oracle, *Java*. [Online] Available: http://www.oracle.com/technetwork/java/javase/downloads/index.html Accessed on: September 5, 2008.

[18] R. Gentleman and R. Ihaka, *R Project for Statistical Computing*. [Online] Available: https://www.r-project.org/ Accessed on: February 8, 2012.

[19] P. Clark, "Elementary school science and math tests as a driver for AI: Take the Aristo Challenge!" in *AAAI*, January 2015, pp. 4019–4021.

[20] P. Clark and N. Balasubramanian, "Interpreting and reasoning with simple natural language sentences," 2014, unpublished.

[21] P. Blackburn, J. Bos, and K. Striegnitz, "Facts, rules, and queries," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 1, pp. 1–20.

[22] P. Blackburn, J. Bos, and K. Striegnitz, "Unification and proof search," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 2, pp. 21–46.

[23] P. Blackburn, J. Bos, and K. Striegnitz, "Recursion," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 3, pp. 47–70.

[24] P. Blackburn, J. Bos, and K. Striegnitz, "Lists," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 4, pp. 71–88.

[25] P. Blackburn, J. Bos, and K. Striegnitz, "Arithmetic," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 5, pp. 89–104.

[26] P. Blackburn, J. Bos, and K. Striegnitz, "More lists," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 6, pp. 105–118.

[27] N. S. Friedland, P. G. Allen, G. Matthews, M. Witbrock, D. Baxter, J. Curtis, B. Shepard, P. Miraglia, J. Angele, S. Staab *et al.*, "Project Halo: Towards a digital Aristotle," *AI magazine*, vol. 25, no. 4, pp. 29–48, Winter 2004.

[28] J. Fan, A. Kalyanpur, D. Gondek, and D. A. Ferrucci, "Automatic knowledge extraction from documents," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 5:1–5:10, May-June 2012.

[29] D. I. Moldovan, S. M. Harabagiu, M. Paşca, R. Mihalcea, R. A. Goodrum, and C. R. Gîrju, "LASSO: A tool for surfing the Answer Net," in *Proc. TREC-8*, 1999, pp. 175–184.

[30] X. Li and D. Roth, "Learning question classifiers," in *Proc. 19th Int. Conf. Computational Linguistics-Volume 1.* Association for Computational Linguistics, 2002, pp. 1–7.

[31] M. H. Clark, "Cognitive illusions and the lying machine: A blueprint for sophistic mendacity," Ph.D. dissertation, Dept. Cog. Sci., Rensselaer Polytechnic Inst., Troy, NY, 2010.

[32] B. S. Firozabadi and Y.-H. Tan, "Formal models of fraud," in *Proc. 4th Int. Workshop Deontic Logic Computer Science (EON'98)*, 1998, pp. 371–385.

[33] J. Johnson, "Toward an intelligent agent for fraud detection – the CFE Agent," in *Deceptive and Counter-Deceptive Machines: Papers from the AAAI Fall Symposium*, no. FS-15-03. AAAI Press, 2015, pp. 21–26.

[34] Association of Certified Fraud Examiners, *2011 ACFE Study Package.* [Online] Available: http://www.acfe.com/student-package.aspx Accessed on: November 1, 2011.

[35] Apache Software Foundation, *Lucene.* [Online] Available: https://lucene.apache.org/ Accessed on: November 12, 2015.

[36] Apache Software Foundation, [Online] Available: https://www.apache.org/ Accessed on: July 15, 2016.

[37] M. McCandless, E. Hatcher, and O. Gospodnetic, "Meet Lucene," in *Lucene in Action, Second Edition: Covers Apache Lucene 3.0.* Greenwich, CT, USA: Manning Publications, 2010, ch. 1, pp. 3–30.

[38] M. McCandless, E. Hatcher, and O. Gospodnetic, "Building a search index," in *Lucene in Action, Second Edition: Covers Apache Lucene 3.0.* Greenwich, CT, USA: Manning Publications, 2010, ch. 2, pp. 31–73.

[39] M. McCandless, E. Hatcher, and O. Gospodnetic, "Adding search to your application," in *Lucene in Action, Second Edition: Covers Apache Lucene 3.0.* Greenwich, CT, USA: Manning Publications, 2010, ch. 3, pp. 74–109.

[40] M. McCandless, E. Hatcher, and O. Gospodnetic, "Lucene's analysis process," in *Lucene in Action, Second Edition: Covers Apache Lucene 3.0.* Greenwich, CT, USA: Manning Publications, 2010, ch. 4, pp. 110–151.

[41] E. Alpaydin, "Supervised learning," in *Introduction to Machine Learning.* Cambridge, MA, USA: MIT Press, 2014, ch. 1, pp. 21–48.

[42] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Lab: Logistic regression, LDA, QDA, and KNN," in *An Introduction to Statistical Learning.* New York, NY, USA: Springer, 2013, section 4.6, pp. 154–167.

[43] S. Bringsjord and N. S. Govindarajulu, "Toward a Modern Geography of Minds, Machines, and Math," in *Philosophy and Theory of Artificial Intelligence*, ser. Studies in Applied Philosophy, Epistemology and Rational Ethics, V. C. Müller, Ed. New York, NY: Springer, 2013, vol. 5, pp. 151–165.

[44] K. Arkoudas and S. Bringsjord, "Propositional Attitudes and Causation," *International Journal of Software and Informatics*, vol. 3, no. 1, pp. 47–65, 2009.

[45] S. Bringsjord, J. Taylor, A. Shilliday, M. Clark, and K. Arkoudas, "Slate: An argument-centered intelligent assistant to human reasoners," in *Proc. 8th Int. Workshop on Computational Models of Natural Argument (CMNA 8)*, F. Grasso, N. Green, R. Kibble, and C. Reed, Eds., Patras, Greece, July 2008, pp. 1–10.

[46] M. E. Stickel, "SNARK - SRI's New Automated Reasoning Kit," 2008, [Online] http://www.ai.sri.com/~stickel/snark.html Accessed on: March 10, 2012.

[47] A. Kolmogorov and V. Uspenskii, "On the Definition of an Algorithm," *Uspekhi Matematicheskikh Nauk*, vol. 13, no. 4, pp. 3–28, 1958.

[48] S. Bringsjord and N. S. Govindarajulu, "In defense of the unprovability of the church-turing thesis," *Journal of Unconventional Computing*, vol. 6, no. 5, pp. 353–373, 2011.

[49] Association of Certified Fraud Examiners, "Doctor/ER Shopping" in *2011 Fraud Examiners Manual*, section 1, p. 1.227, Austin, Texas: Association of

Certified Fraud Examiners [Online] Available: http://www.acfe.com/2016USFEM.aspx Accessed on: November 1, 2011.

[50] P. Blackburn and J. Bos, "First-order logic," in *Representation and Inference for Natural Language: A First Course in Computational Semantics (Studies in Computational Linguistics).* Stanford, CA, USA: Center for the Study of Language and Information, 2005, ch. 1, pp. 1–54.

[51] P. Blackburn and J. Bos, "Lambda calculus," in *Representation and Inference for Natural Language: A First Course in Computational Semantics (Studies in Computational Linguistics).* Stanford, CA, USA: Center for the Study of Language and Information, 2005, ch. 2, pp. 55–104.

[52] P. Blackburn and J. Bos, "Underspecified representations," in *Representation and Inference for Natural Language: A First Course in Computational Semantics (Studies in Computational Linguistics).* Stanford, CA, USA: Center for the Study of Language and Information, 2005, ch. 3, pp. 105–154.

[53] P. Blackburn and J. Bos, "Propositional inference," in *Representation and Inference for Natural Language: A First Course in Computational Semantics (Studies in Computational Linguistics).* Stanford, CA, USA: Center for the Study of Language and Information, 2005, ch. 4, pp. 155–202.

[54] P. Blackburn and J. Bos, "First-order inference," in *Representation and Inference for Natural Language: A First Course in Computational Semantics (Studies in Computational Linguistics).* Stanford, CA, USA: Center for the Study of Language and Information, 2005, ch. 5, pp. 203–258.

[55] P. Blackburn and J. Bos, "Putting it all together," in *Representation and Inference for Natural Language: A First Course in Computational Semantics (Studies in Computational Linguistics).* Stanford, CA, USA: Center for the Study of Language and Information, 2005, ch. 6, pp. 259–308.

[56] P. Blackburn, J. Bos, and K. Striegnitz, "Definite clause grammars," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 7, pp. 119–138.

[57] P. Blackburn, J. Bos, and K. Striegnitz, "More definite clause grammars," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 8, pp. 139–158.

[58] P. Blackburn, J. Bos, and K. Striegnitz, "A closer look at terms," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 9, pp. 159–184.

[59] P. Blackburn, J. Bos, and K. Striegnitz, "Cuts and negation," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 10, pp. 185–202.

[60] P. Blackburn, J. Bos, and K. Striegnitz, "Database manipulation and collecting solutions," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 11, pp. 203–216.

[61] P. Blackburn, J. Bos, and K. Striegnitz, "Working with files," in *Learn Prolog Now!* London, England: College Publications, 2006, ch. 12, pp. 217–230.

[62] J. Martin and D. Jurafsky, "Statistical parsing," in *Speech and Language Processing. An introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed. Saddle River, NJ, USA: Prentice Hall, 2000, ch. 14, pp. 459–488.

[63] Apache Software Foundation, *OpenNLP*. [Online] Available: https://opennlp.apache.org/ Accessed on: August 14, 2015.

[64] Stanford Natural Language Processing Group, *Stanford Parser*. [Online] Available: http://nlp.stanford.edu/software/lex-parser.shtml Accessed on: August 14, 2015.

[65] "Standard Boolean model — Wikipedia, The Free Encyclopedia" in *Wikipedia*, [Online] Available: https://en.wikipedia.org/w/index.php?title=Standard_Boolean_model Accessed on: February 1, 2016.

APPENDIX A
Brief Overview of Information Retrieval

Information Retrieval (IR) [4] is the branch of AI dealing with the creation of software that retrieves documents of an unstructured nature from among a large collection of such documents in response to an information need expressed as a natural language query. A document is a unit of data, typically unstructured or semi-structured, and typically expressed in natural language. (One of the fundamental questions to consider in IR is how to define a document - a sentence? a paragraph? a chapter? The answer typically depends on the nature of the problem domain.) A document collection is simply a collection of such documents, as defined above. Finally, a vocabulary, $V = \{t_1, t_2, t_3, ..., t_n\}$ is the set of all terms over which the contents of the documents are defined.

A.1 Boolean Retrieval

Boolean information [4] retrieval is based on the idea of dividing up the document collection into two sets for each query - one set of documents which meets the requirements of the boolean query and the other set whose documents does not. Boolean queries are structured as conjunctions of disjunctions; that is, of the form of query, q, where, as given in [65],

$$q = (W_i \vee W_k \vee ...) \wedge ... \wedge (W_j \vee W_s \vee ...) \tag{A.1}$$

where $W_i = t_i, W_k = t_k, W_j = t_j, W_s = t_s$, or $W_i = NON\ t_i, W_k = NON\ t_k, W_j = NON\ t_j, W_s = NON\ t_s$ and where t_i means that t_i exists in the document and $NON\ t_i$ means it does not .

A.1.1 Term-Document Incidence Matrix

Boolean retrieval may be implemented using a data structure called a term-document incidence matrix [4], A, where A is a two-dimensional array in which the ith row denotes the ith document in the document collection and where jth column

denotes the jth term in the vocabulary, and where $A[i, j] = 1$ if the term, i exists in the document, j, and 0 otherwise. Based on A, the processing of a query involves finding those documents for which there is a 1 in each of the rows corresponding to the terms of the query [4].

A significant pitfall of this method is that it requires a vast amount of memory. Consider an example, as explained in [4], consisting of 1 million documents and a vocabulary of 500,000 words. Then, the size of the matrix is 50 trillion (1 million x 500,000). This matrix is also highly sparse since any given document has on average a small number of words relative to the size of the vocabulary. Suppose, for example, each document has 1,000 words. Then, for each document, among the 50,000 elements in its corresponding row of the matrix, only 1,000 (2%) are non-zero.

A.1.2 Inverted Index

An alternative implementation for boolean retrieval utilizes an inverted Index [4] – a hash table in which the keys are the terms in the vocabulary and the value is a linked list of all of the identifiers for those documents in which that term appears. This technique exploits the sparsity of the term-document matrix – requiring memory only for those elements in which $A(i, j) = 1$ [4].

A.2 Ranked Retrieval

Boolean retrieval has the unfortunate drawback of returning a set of documents in response to a query as a set of equally ranked units through which the user must sift in order to find the information sought [7]. Sometimes, this sifting can be a sizable task depending on the number of documents returns from the boolean query. Ranked retrieval, on the other hand, is IR in which documents are ranked according to a score that measure the degree of similarity between the query and the terms of the document and in which documents are returned in decreasing sorted order of this score [7].

The Vector Space model (VSM) [7] is one approach to ranked retrieval, and is based on representing the documents in the collection and the query as vectors and where documents are scored according to a measure of similarity between their

vector representations and that of the query. There are a number of measures, or weights, that are typically incorporated into a vector representation as discussed below.

The log of the term frequency [7], $\log_{10}(tf)$ is a measure of the number of occurences of a particular term in the document. (The log is used as opposed to the term frequency, itself, in order to dilute the effect for each additional occurrence of a term.) As explained in [7], the log-frequency weight of term, t in document, d is given by:

$$w(t,d) = 1 + \log_{10}(tf_d), \text{where } tf_d > 0, 0 \text{ otherwise} \tag{A.2}$$

Inverse document frequency, as explaned in [7], is a measure of the relative scarcity of a term in a document relative to the other documents in the collection. A higher weight is assigned to a term that appears in relatively few documents compared with one that appears in many. As explained in [7], the inverse document frequency, $idf(t)$, can be calculated for each term, t, of the query as follows:

$$idf(t) = \log_{10}(N/df_t), \tag{A.3}$$

where N is the number of documents in the collection and df_t is the number of documents containing term, t. Again, as is the case for term frequency, the log is used here to moderate the effect of the measure.

We can combine these two measures such that for each term, t, in each document, d, we have, as explained in [7]:

$$w(t,d) = (1 + \log_{10}(tf_d)) \times \log_{10}(N/df_t). \tag{A.4}$$

The vector representation of a document, d, is $\vec{d}$, where, as given in [7],

$$\vec{d} = [w(t_1,d), w(t_2,d), w(t_3,d), ..., w(t_n,d)], \tag{A.5}$$

where, as defined above, our vocabulary, $V = \{t_1, t_2, ..., t_n\}$. Note, queries can also be vectorized in the same way.

In order to measure the relative relevance of document, d, to the query, q, the vector representations of both d and q, $\vec{d}$ and $\vec{q}$, are harnessed to compute a quantitative measure of the level of similarity between the two vectors using the concept of cosine similarity [7]. Informally, the cosine similarity of two n-dimensional vectors is a measure of the cosine of the angle between them in n-dimensional space. Normalization of the lengths of the vectors is also incorporated into this calculation to assure that longer documents' weights do not outsize the weights of shorter (but perhaps, similar) documents, (such as the query itself, which is commonly much shorter than the document against which it is compared). Thus, as explained in [7], we have

$$\begin{aligned} \text{sim}(q,d) &= \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} \\ &= \frac{\vec{q}}{|\vec{q}|}\cdot\frac{\vec{d}}{|\vec{d}|} \end{aligned} \tag{A.6}$$

For a document, d whose length-normalized-vector representation is similar to that of q, the angle between its vector and that of q should be small (close to 0), and thus have a cosine near 1. For those documents not similar to q, the cosine measure will tend toward 0, (note that all of the terms in these vector-representation vectors are greater than or equal to 0). Under the document vectorization approach, the highest K documents are returned in response to a query, q, in order of decreasing cosine similarity, where K is an arbitrary figure intended to limit the size of the return set.