JOURNAL OF INFORMATION SYSTEMS
Vol. 19, No. 1
Spring 2005
pp. 19–41

Financial Reporting and Auditing Agent with Net Knowledge (FRAANK) and eXtensible Business Reporting Language (XBRL)

Matthew Bovee
The University of Vermont

Alexander Kogan
Rutgers, The State University of New Jersey

Kay Nelson
The Ohio State University

Rajendra P. Srivastava
The University of Kansas

Miklos A. Vasarhelyi
Rutgers, The State University of New Jersey

ABSTRACT: This paper describes the development and applications of FRAANK—Financial Reporting and Auditing Agent with Net Knowledge. The prototype of FRAANK presented here provides automated access to, and understanding and integration of, rapidly changing financial information available from various sources on the Internet. In particular, FRAANK implements intelligent parsing to extract accounting numbers from natural-text financial statements available from the SEC EDGAR repository. FRAANK develops an "understanding" of the accounting numbers by means of matching the line-item labels to synonyms of tags in an XBRL taxonomy. As a result, FRAANK converts the consolidated balance sheet, income statement, and statement of cash flows into XBRL-tagged format. Based on FRAANK, we propose an empirical approach toward the evaluation and improvement of XBRL taxonomies and for identifying and justifying needs for specialized taxonomies by assessing a taxonomy fit to the historical data, i.e., the quarterly and annual EDGAR filings. Using a test set of 10-K SEC filings, we evaluate FRAANK's performance by estimating its success rate in extracting and tagging the line items using the year 2000 C&I XBRL Taxonomy, Version 1. The evaluation results show that FRAANK is an advanced research prototype that can be useful in various practical applications. FRAANK also integrates the accounting numbers with other financial information publicly available on the Internet, such as timely stock quotes and analysts' forecasts of earnings, and calculates important financial ratios and other financial-analysis indicators.

Keywords: semantic pattern matching; SEC EDGAR; XBRL taxonomy.

I. INTRODUCTION

The accounting world has changed dramatically over recent years and the pace of change is accelerating. Major accounting firms have increasingly been using technology to obtain labor savings. They are progressively developing large knowledge-management services to support their professional activities. Input for such knowledge-management services comes from many valuable data sources that are publicly available on the Internet. The most notable one is the Securities and Exchange Commission's EDGAR (Electronic Data Gathering, Analysis, and Retrieval) Internet repository containing corporate filings with the SEC (http://www.sec.gov/edgarhp.htm). Many Internet portals (e.g., Yahoo!, Quicken) provide public access to financial markets data, analysts' forecasts, news feeds of business relevance, etc.

Numerous professionals depend critically on timely access to financial information. Complexity, cost, and latency of obtaining financial information in computer-understandable format create significant friction in the system. Although the SEC filings are available from EDGAR in a computer-readable format, they are difficult to process automatically because of great variations in the filing structure and terminology, which impede the automatic understanding of semantics of plain-text documents. A very promising approach to solving this problem is based on the development and wide acceptance of XML (eXtensible Markup Language). XBRL (eXtensible Business Reporting Language)[1] is a recent XML-derivative language aimed at enhancing and facilitating the business reporting and analysis process at the account level. Several other XML-derivative languages (such as ebXML), proposed in different industries or by associations, focus on the transaction level. While the development of XBRL shows great promise, there are significant barriers to wide acceptance of XBRL, including the costs associated with adopting XBRL and the absence of the SEC requirement to file financial statements in XBRL. Another major impediment to widespread XBRL adoption is the difficulty of agreeing on common taxonomy standards. Before XBRL becomes the accepted standard for filing financial statements, intelligent software tools must be developed and utilized to automatically process the semi-structured natural-text financial documents that are publicly available on the Internet. Manual extraction of accounting numbers from the text of accounting statements available from EDGAR and manual conversion to the XBRL standard would be too cumbersome and expensive.

The objective of this paper is to describe the development, evaluation, and applications of FRAANK—Financial Reporting and Auditing Agent with Net Knowledge. The prototype of FRAANK presented here provides automated access to, and understanding and integration of, rapidly changing financial information available from various sources on the Internet. In particular, FRAANK implements intelligent parsing to extract and understand accounting numbers from natural-text financial statements available from EDGAR, and converts the consolidated balance sheet, income statement, and statement of cash flows into XBRL-tagged format. It integrates the accounting numbers with other financial information publicly available on the Internet, such as timely stock quotes and analysts' forecasts of earnings, and calculates important financial ratios and other financial analysis indicators. FRAANK can potentially be utilized in an auditing practice, or used by investors and creditors in decision making.

[1] See http://www.xbrl.org.

More specifically, FRAANK is focused on the task of finding important information (e.g., accounting numbers) in semi-structured natural-text documents of accounting or auditing nature. The pioneering research in developing artificial-intelligence capabilities for analyzing financial statements and understanding accounting texts (Tanaka 1982) is described by Mui and McCarthy (1987), O'Leary and Munakata (1988), O'Leary and Eis (1991), and O'Leary and Kandelin (1992). The most comprehensive previously reported implementation of the intelligent parsing of financial statements in the SEC EDGAR filings has been achieved in EdgarScan (Ferguson 1997; Steier 1995; Steier et al. 1997). A generic tool for semantic parsing of the SEC filings has been developed by Gerdes (2003). The predecessor of FRAANK, the so-called original EDGAR agent (Nelson et al. 2000), was capable of retrieving quarterly filings (10-Qs) from the SEC EDGAR repository and parsing them to identify the most important accounting numbers. As compared with the new EDGAR subagent of FRAANK, the original EDGAR agent was much more limited in its capabilities and had a much simpler design. The original EDGAR agent could process only the quarterly SEC filings and identify only a few of the most important accounting numbers. Its design was specially tuned for this limited task, and could not be scaled up to parse financial statements comprehensively. It interacted with a single online information source (the SEC EDGAR repository) only.

EdgarScan was created by PwC (2001) and made publicly available on the Internet in response to the need for automatic extraction of accounting numbers from the EDGAR filings.[2] Although EdgarScan is technologically very different from the original EDGAR agent (it uses C and Prolog instead of Perl), its architecture and design have exactly the same limitations as those of the original EDGAR agent. EdgarScan is designed to parse a predetermined limited set of accounting numbers, and therefore cannot be easily scaled up to understand the meaning of the financial statement labels that correspond to the lower levels of an XBRL taxonomy. Making EdgarScan (and other similar parsers) understand all the line items in the financial statements would require a total redesign of the architecture, similar to the effort reported in this paper, which involved transforming the original EDGAR agent into the new version of FRAANK.

The new version of FRAANK is capable of parsing both the annual (10-K) and quarterly (10-Q) SEC filings and is designed to ''understand'' all the line items in the consolidated financial statements by matching them to the most appropriate tags in an XBRL taxonomy. The architecture of FRAANK facilitates learning over time since it makes it possible to improve FRAANK's accuracy by adding synonyms to the knowledge base whenever an exception is logged during execution. FRAANK's use of an XBRL taxonomy as the anchor for matching the line items makes it a natural tool for empirically evaluating various XBRL taxonomies, identifying their deficiencies for improvement, and justifying the need for additional industry taxonomies.[3] The utilization of FRAANK as a converter of regular SEC filings to XBRL can facilitate automation of various financial analyses that previously had to rely on human intervention. Such uses are further enhanced by FRAANK's capabilities of extracting additional financial information online, such as stock quotes and analysts' forecasts.

---

[2] EdgarScan is a web-based interface to the SEC EDGAR filings, which is driven by a back-end engine that regularly retrieves EDGAR filings from the SEC servers and automatically parses them to find a subset of key accounting numbers, normalizes them to a common format, and stores the results in its own database. (See the homepage of EdgarScan at http://edgarscan.pwcglobal.com/servlets/edgarscan.)

[3] While the possibility of using FRAANK to evaluate XBRL taxonomies is a ''side benefit'' of FRAANK's architecture, this capability will become more important as the development of various XBRL taxonomies comes to the forefront of financial reporting.

This paper is organized as follows. In the next section we outline the importance and various issues related to XBRL as the emerging business-reporting language and the need for intelligent agents to facilitate the adoption of XBRL. Then, in Section III we define the goals of FRAANK, and describe its function and architecture. In Section IV we give some technical details about the implementation and intelligent parsing in FRAANK. In Section V we evaluate the performance of FRAANK. We devote Section VI to describing various applications of FRAANK, and Section VII to discussing some of the key future developments in the FRAANK project and concluding remarks.

II. BACKGROUND

XBRL has come a long way since the pioneering development of its predecessor XFMRL (eXtensible Financial Reporting Markup Language) in 1999. The XBRL Consortium, which includes numerous interested parties such as CPA firms, business-software vendors, and financial-services companies, has been continuously working on developing and improving various components of this emerging standard. The main focus of XBRL has been on developing means for unambiguously identifying financially relevant information. As an XML-based standard, XBRL is designed to provide a set of textual tags for marking various parts of a document to identify accounting numbers relevant for external financial reporting. Adding such markup makes it possible to automate the processing of financial reports and facilitates the exchange of financial information between various computer systems.

To accommodate significantly diverging financial reporting practices among different countries and various industries, XBRL uses a two-tiered design consisting of the XBRL specification and XBRL taxonomies. The XBRL specification provides a set of generic tags for expressing various facts and, as such, the XBRL specification is not tied to any particular set of accounting standards or practices, or even to accounting in general. To provide concrete tags for marking an actual financial report, the XBRL specification has to be complemented with an XBRL taxonomy. It is an XBRL taxonomy that defines the markup of various accounting concepts as well as the hierarchical and numerical relationships among these concepts. Therefore, while a single XBRL specification suffices, numerous taxonomies must be developed to accommodate the needs of different countries and/or industries.

Developing XBRL taxonomies is a difficult, laborious, and controversial process. A typical taxonomy has to define markup for several hundred concepts, and it has to be done in accordance with generally accepted accounting standards and practices. The standards are usually not formal enough and the practices are not uniform enough to allow for an unambiguous translation of them into taxonomies. On the one hand, it is important to develop a sufficient number of taxonomies to accommodate the true needs of all the constituents. On the other hand, it is important to avoid the proliferation of different taxonomies to the extent possible, since the use of different taxonomies impedes the comparability of financial reports. It is therefore important to develop a methodology for evaluating proposed XBRL taxonomies, developing ways of improving them, and identifying and justifying needs for additional taxonomies.

An empirical approach toward the evaluation and improvement of XBRL taxonomies is to assess a taxonomy fit to the historical data, i.e., the quarterly and annual filings with the SEC that are available from the EDGAR repository. The FRAANK agent can be used to automate the evaluation and improvement of an XBRL taxonomy, which is utilized in FRAANK as the anchor in intelligent parsing of the natural text of financial statements. Most importantly, FRAANK facilitates the adoption of XBRL by providing the automatic

conversion of financial statements to this emerging standard. Thus, FRAANK demonstrates the crucial importance of the intelligent-agent technology for the successful development and adoption of XBRL.

While software agents have been the focus of much attention both in popular press and in the research community, there is no commonly accepted definition of an agent.[4] Past experience with various artificial-intelligence applications has shown that to be successful in practice an application usually has to follow the ''weak-AI'' paradigm, i.e., to simulate intelligent behavior, and be sufficiently focused on a limited set of tasks.[5] This is the approach taken in FRAANK. The main subagent of FRAANK—EDGAR—faces a difficult task of understanding very loosely formatted financial statements in the SEC filings, and therefore utilizes a tremendous amount of code to achieve performance worthy of the weak-AI qualification. On the other hand, the ticker subagent's task is much simpler: to choose the most likely ticker symbol for a given string representing a company name. While, strictly speaking, this task is ill-defined, the performance of the ticker subagent is often comparable to that of a human being given the same task, and therefore quite useful even though it implements only a few simple heuristic rules (see the next section), and is an extreme example of the weak-AI approach.

III. ARCHITECTURE OF FRAANK

The current version of the FRAANK is capable of:

> analyzing semi-structured natural text in the financial domain,
> formally representing and using accounting knowledge, and
> interacting with a variety of online information sources.

The FRAANK prototype[6] can be viewed as the first step in gathering financial data on companies that are available on the Internet and using these data to provide value-added service. Once the agent retrieves financial information, these data can be processed and combined with other artificial-intelligence systems to provide knowledge for enhanced decision making in real time.

FRAANK communicates with its users over the World Wide Web through a simple interface. The user who is interested in a certain publicly traded company submits that company's name to the agent, and then FRAANK automatically:

> searches the SEC EDGAR database for the filings (10-Q and 10-K) by that company, intelligently parses, and tags using the U.S. C&I XBRL taxonomy (Spec. 2) the filing selected by the user,
> queries the Yahoo! ticker-search engine to identify the ticker of the company,
> uses the ticker to retrieve the most recent stock price from Quote.com,
> contacts Quicken.com to find the most current consensus forecast of earnings per share (EPS) provided by Zacks, and

---

[4] For an extensive discussion of what constitutes an agent, see Franklin and Graesser (1997).

[5] There is an ongoing debate in the AI literature between the proponents of the so-called ''strong AI'' versus ''weak AI'' (see, e.g., Searle 1984). The strong-AI proponents claim that a ''true'' AI program has to be a universal model of a human-thinking process. The proponents of weak AI argue that it is more important and productive to concentrate on designing programs that only simulate intelligent behavior, i.e., produce results similar to what a human thinker would have done. While the weak-AI approach emphasizes that seemingly intelligent behavior may result from a seemingly trivial code, in most cases achieving useful results requires implementing very complex logic.

[6] The prototypes of the FRAANK agent that process the 10-Q and 10-K filings, respectively, can be found at: http://fraank.eycarat.ku.edu/cgi-bin/vml/working/FRAANK/login.htm.

utilizes the obtained results to calculate various accounting ratios, and the Z-score (a discriminant measure of bankruptcy risk) (Altman 1968, 1983).

The multifaceted nature of FRAANK is reflected in its design. FRAANK consists of several subagents, corresponding to the Internet information sources that FRAANK utilizes in real time. These subagents are the EDGAR agent, the ticker agent, the stock-quote agent, and the EPS agent. The architecture of FRAANK, as shown in Figure 1, allows the agent logic to be clearly separated from the end-user interface on the one hand, and from the accounting knowledge source stored in a relational database on the other hand. Currently, the knowledge source in the database contains synonyms of accounting terms.

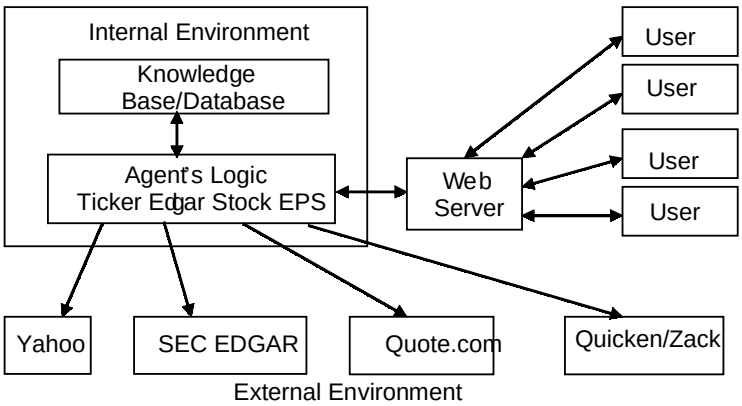
EDGAR Subagent

EDGAR is the main subagent of FRAANK, since it is responsible for obtaining and processing the richest source of information: SEC filings of publicly traded companies available in the SEC EDGAR repository on the Web. The predecessor of the EDGAR subagent, the original EDGAR agent, was described in detail in Nelson et al. (2000). The current version of the EDGAR subagent is completely redesigned, significantly more robust, and has much greater capabilities, which include:

retrieving the company's recent SEC filings,
finding and extracting from them consolidated financial statements,
parsing the statements to identify all the line items, their balances, and their aggregation structure,
matching the labels of the identified line items with the corresponding XBRL tags and tagging them, and
identifying line items for which no XBRL tag in the taxonomy exists.

This subagent searches the SEC EDGAR repository for reports (10-Q and 10-K) of publicly traded companies that match the user-specified names and returns a list of available filings. These filings are sorted by date, with the most recent ones listed first. The user is prompted to choose a filing, which is then retrieved and analyzed. Since the EDGAR subagent always contacts the SEC EDGAR database in real time over the Internet, it always

FIGURE 1
The Architecture of FRAANK



provides access to the most recent filings. While this retrieval function is based on standard Internet technology, the other two functions of EDGAR have to do with extremely difficult and highly nonstandard problems of analyzing natural text.

The current version of FRAANK finds in the retrieved 10-Q and 10-K filings the consolidated balance sheet, income statement, and statement of cash flows. Although a human accountant can easily locate a financial statement in the body of a 10-K filing, the same task is a challenge for a computer program because of the great variability in the structure of the filings and the language used. The location of the statements in the body of the filing varies significantly. While the SEC EDGAR filings are required to have some generic SGML tags such as the table and column tags, evidently there is no effective validation procedure in place to enforce compliance with these requirements, and the required tags can be missing or misplaced. While each statement is delimited with the table tags, there are usually many different tables in the body of the filing. Moreover, the caption can be both before and after the table tag, and the wording of the caption varies greatly. For example, the income statement can also be called the statement of income, the statement of changes in financial position, the profit and loss statement, or the statement of revenues and expenses, etc. Complicating the automatic location of statements even further is the possibility of having the same keywords in the caption of some other tables in the body of the filing (e.g., due to the segment reporting requirements).

After a particular financial statement is extracted, it is parsed by the EDGAR subagent to identify all the line items, their balances, and their aggregation structure. Line-item names can exhibit even greater variability than the table captions, as described in detail in Nelson et al. (2000). The intelligent parsing procedure in EDGAR utilizes tables of line-item synonyms stored in a relational database (see Section IV for a more detailed description). It used to be possible to cross-validate certain line items with the SGML-tagged values presented in the Financial Data Schedule (FDS) appended to the 10-K and 10-Q filings. Unfortunately, the SEC abandoned the requirement to append the FDS as of early 2001, thus making this cross-validation impossible for newer filings. Therefore, for recent filings, only the aggregation structure (totals, subtotals, etc.) of the line numbers is used to improve the accuracy of parsing.

Once the agent has parsed the SEC filings and identified the line items and the corresponding values for the desired period, it matches the identified labels to the corresponding XBRL tags and then tags the line items accordingly. The current version of the agent uses the 2000 U.S. C&I XBRL Taxonomy, Version 1 (see XBRL 2000). Whenever there are no matching XBRL tags, the agent uses generic tags for those line items.[7] As discussed later, this feature of the agent will be utilized in testing, validating, and improving the US C&I XBRL taxonomies.

Ticker Subagent

After the user selects a 10-K or 10-Q filing for analyses, FRAANK uses its ticker subagent to identify the ticker symbol of the company, which is needed for obtaining the stock quotes and EPS forecasts. There are a number of ticker-search engines publicly available on the Internet. The ticker agent sends the company name to the Yahoo! ticker engine, and retrieves a list of tickers matching the submitted name. The retrieved ticker list must

[7] For example, the balance sheet of Compaq in its 2001 10-K filing has a line item labeled "Leases and other accounts receivable." Since the XBRL taxonomy does not have an appropriate tag, the agent tags this item with a generic tag "KU:unknown" and labels it with the correct description.

then be analyzed to determine the most probable candidate, since this list will usually contain quite a few tickers. The ticker subagent utilizes several heuristic rules for identifying the best match, applying them in the following order:

> Since most probable matches should include user input as separate words in company names, items not containing all the exact words (e.g., containing super-words, i.e., words that contain the input words as proper substrings) are eliminated.
> Since most probable ticker symbols are short, the list is ordered by ticker-symbol length.
> Since most probable company names are short, items corresponding to the shortest tickers are ordered by the total length of the company name, and the first one is chosen as the best match. (Any tie is broken arbitrarily at this stage.)

Although these rules are quite simple, and thus the ticker subagent intelligence is rudimentary, very often the resulting best match is exactly the right ticker. This subagent illustrates the practicality of simple heuristics: certain seemingly complicated tasks can be accomplished by using quite simple means. Unfortunately, as the example of the EDGAR subagent shows, this is not always the case (see Section IV for a description of our intelligent parsing technology).

Stock Quotes and EPS Subagents

The stock-quote subagent uses the ticker symbol identified by the ticker subagent to send the symbol to a public-stock-quote engine (mach.quote.com), and extracts from the response the most recent share price. On its Quicken website, Intuit provides free access to Zacks Investment Research data of earnings-per-share (EPS) forecasts. The EPS subagent uses the most probable ticker to retrieve the (diluted) EPS forecast for the next quarter from Quicken.com.

Integration and Analysis

FRAANK integrates the subagents' responses (price from the stock-quote subagent, earnings from EDGAR, and analyst forecasts) to compute historical and estimated important financial ratios showing this company's investment potential (like the price/earnings ratio), and to calculate a number of other financial ratios (e.g., quick ratio, current ratio). Examples of summarizing and integrating the information obtained by the subagents are detailed in Section VI.

IV. DESIGN, IMPLEMENTATION AND INTELLIGENT PARSING IN FRAANK

Since the most difficult task of FRAANK is the intelligent analysis of natural-text documents (i.e., 10-K and 10-Q filings), FRAANK needs a very strong pattern-matching capability. Therefore, the programming logic of FRAANK is implemented entirely in Perl (Wall et al. 1996) since, arguably, Perl provides one of the strongest pattern-matching functionalities. By far the largest part of FRAANK's code is devoted to the implementation of intelligent parsing of semi-structured natural text of the SEC filings.

Since FRAANK interacts with both its users and its external information sources over the Internet, the choice of Perl provides the benefit of very strong networking support. When FRAANK communicates with external information sources over the Internet (e.g., SEC, Yahoo!, Quicken, etc.), FRAANK essentially acts as a specialized web client. The web client functionality of FRAANK is implemented using the high-level libwww-perl

library (2001), once called LWP (see also Wong 1997), which allows much simpler implementation of more sophisticated web client functionality in the current version of FRAANK (as compared with the original EDGAR agent, see Nelson et al. 2000).

End-users also interact with FRAANK using HTTP. In this case, however, the user's web client contacts FRAANK's web server, which launches FRAANK using Common Gateway Interface (CGI). FRAANK relies on the CGI.pm module developed by Stein (1999), which uses objects to create web forms on the fly and to parse their contents. When executed by the HTTP server, the agent receives the user's input as CGI URL-encoded variables using the CGI.pm param() method.

The EDGAR subagent of FRAANK has grown and transformed greatly since the original implementation (described in Nelson et al. [2000]). It has been redesigned and modularized, and now includes almost two dozen subroutines and about 5,000 lines of code. The major subroutines are designed for:

- extracting the consolidated balance sheet, income statement, and statement of cash flows;
- partitioning each statement into line items, identifying headings, and merging multiple lines corresponding to the same line item;
- splitting each line item into the label and values and identifying the column of values corresponding to the current period (as opposed to the previous ones);
- determining the aggregation structure of the values to identify totals among the line items;
- matching the labels to the synonyms of tags in an XBRL taxonomy and selecting the most appropriate XBRL tag for each line item; and
- matching the headings to the identified totals, and selecting the most appropriate XBRL tags for the totals with empty labels.

Note that the first three tasks in the above list become much easier if the filings are presented in HTML (as opposed to plain text with a few SGML tags). Therefore, the subagent determines whether the filing is EDGAR or is not in HTML, and chooses the parsing procedures accordingly. The parsing procedures in the EDGAR subagent are heuristic algorithms that simulate the results of information processing by an intelligent reader of the SEC filings. To achieve a high level of accuracy requires implementing a tremendous amount of parsing logic in the code of the EDGAR subagent. For example, the extraction of tables of the three consolidated financial statements is based not only on locating the table tags and recognizing the captions corresponding to the synonyms of the statement titles stored in the knowledge base, but also on verifying that the beginning of the analyzed table corresponds to the expected structure. In the case of the balance sheet, the table is expected to start with assets, more precisely the most liquid assets (cash and cash equivalents). The income statement is expected to start with revenues or sales, while the statement of cash flows is expected to start with net income (for the statements prepared using the indirect method) or cash flows from operations. Additional logic is implemented to identify the end of the statement, which can be split into several parts (e.g., when it does not fit on a single page).

A special challenge in parsing the financial statements is the presence of headings and totals. Headings will usually have just the label and no values, while totals will often have just a trivial label (e.g.,"total") and meaningful values. A heading may or may not have a corresponding total below in the statement, while a total may or may not have a corresponding heading above in the statement. The identification of totals cannot be based only

on matching keywords (such as ''total'') and on locating (double) underlining since both can be missing. Therefore, the procedure for identifying the totals without meaningful labels is based on identifying the aggregation structure of the accounting numbers in the statement, under the assumption that spurious equalities are highly unlikely. *The totals procedure* is recursive. It starts from the top of the statement, adds up the values of contiguous segments of line items, and checks whether the sum equals the value of the following line item. If yes, then the following line item is identified as a total and only this item is used instead of the corresponding segment for further identification of the nested totals. The procedure has to restart after each total is identified. Note that the maximum number of contiguous segments is only quadratic in the number of line items in the statement, and therefore the procedure is not very expensive computationally.

The totals procedure also incorporates special logic for identifying negative values, which is quite challenging since many filings do not follow the standard accounting convention of displaying negative values in parentheses, nor do they include any keywords (such as ''less'') that indicate negative values. Therefore, the procedure attempts to flip the sign of some values in the segment while calculating the total. Note that it would be prohibitively expensive computationally to attempt flipping the sign of every conceivable subset of values in the segment, and one can argue that in general it is computationally intractable to identify an arbitrary subset of negative values.[8] Therefore, it is assumed that the subset of the negative values will form a contiguous subsegment at the end of the considered segment (just before the total), and the procedure attempts to consider all such subsegments, which is computationally inexpensive since the number of such subsegments is at most $n - 1$, where n is the number of values in the segment.

To ''understand the meaning'' of the line items, the EDGAR subagent has to map every line item (if possible) to a tag in an XBRL taxonomy. As illustrated in Figure 2, the EDGAR subagent utilizes a knowledge source of accounting synonyms, stored in a relational database, to cope with the variation encountered in corporate use of terminology and subsequently identify and parse the appropriate substitute terms. The knowledge source is based on an XBRL taxonomy, with each taxonomy tag mapped to (possibly many) synonyms encountered in previously analyzed SEC filings. The connection to the database is established over ODBC, and the queries are implemented in standard SQL.

In the current implementation of the EDGAR subagent, a unified parsing subroutine is developed that parses all the line items in exactly the same way. The 552 elements of the C&I XBRL taxonomy, for the balance sheet, income statement, and statement of cash flows (see Table 1), are used as the key items in a growing database of (currently) approximately 3,200 synonyms. This development is likely to ultimately both improve the accuracy of FRAANK's parsing and to identify possible deficiencies in, or necessary extensions to, the XBRL taxonomy. Indeed, the agent keeps track of the exceptions whenever the parsing subroutine fails, which can happen for two different principal reasons. The first one happens when no synonym in the database matches the line-item description. After human review of the exceptions log, this synonym will be added to the database with its paired corresponding key, thus eliminating this instance of failure in the future. The second reason for a parsing failure is the absence of the corresponding key item in the database of synonyms,

[8] Such problems are known as ''NP-hard'' in the theory of computational complexity (see, e.g., Garey and Johnson 1979).

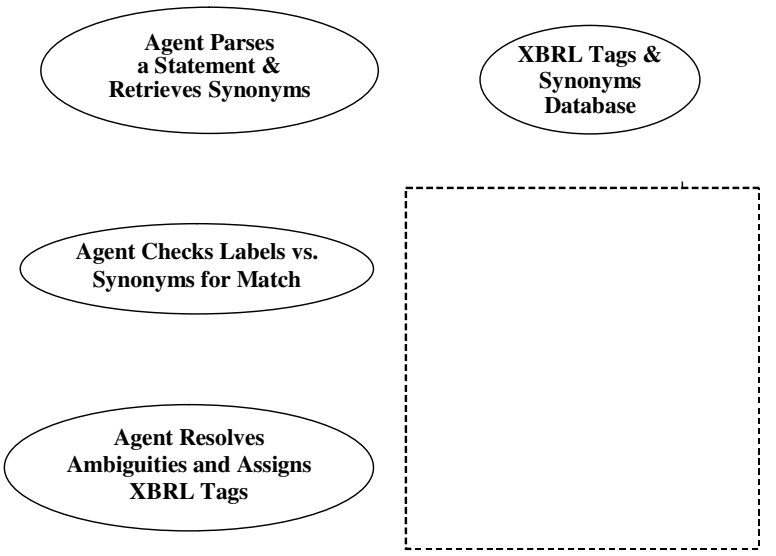



FIGURE 2
Synonym Matching in FRAANK