REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-02-

0411

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching
the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions
Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TY
		12 OCT 01 TO 11 OCT 02

4. TITLE AND SUBTITLE
INTELLIGENT RECORD LINKAGE TECHNIQUES BASED ON INFORMATION RETRIEVAL, NATURAL LANGUAGE PROCESSING, AND MACHINE LEARNING

5. FUNDING NUMBERS
F49620-01-C-0055

6. AUTHOR(S)
ELIOT LI

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SCIENTIFIC SYSTEMS COMPANY, INC.
500 WEST CUMMINGS PARK, SUITE 3000
WOBURN, MA 01801

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
4015 Wilson Blvd, Room 713
Arlington, VA 22203-1954

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-01-C-0055

11. SUPPLEMENTARY NOTES

20030106 104

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The objective of this STTR project is to develop an information management system to rapidly and accurately linking records of related information from web-based information sources. The sheer magnitude of information available online via the Internet has overwhelmed the ability of existing search tools to produce useful query responses. Current web-search techniques typically fail to correlate relevant documcnts that are identified in different ways, such as synonyms and acronyms (aliases). The challenge is to find an approach that can obtain highly accurate matches even when those documcnts do not share any obvious attributes with the query, and with minimal information requirement from the user. Latent Semantic Analysis (LSA) is a technique for identifying both semantically similar words and semantically similar documents. On the face of it, LSA should work well for the task of discovering aliases. That is, for a given word we can use LSA to produce a rank-ordered list of words that are semantically similar to it and aliases for the name should be high in this list. In this Phase I, wc tcstcd this conjecture empirically and found, surprisingly, that under a broad range of circumstances a straightforward application of LSA fails to rank true aliases highly. We then developed a two-stage algorithm that takes the output of LSA, creates a new set of pseudo-documents, and runs LSA again on these new documents. Empirical results show that this two-stage algorithm performs remarkably well in identifying aliases, even in those cases for which a single application of LSA fails miserably. University of Maryland (Baltimore County) is the research institute partner for this effort, under the direction of Professor Charles Nicholas and Tim Oates.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	40
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

SSCI Project# 1322

Nov. 11, 2002

Scientific Systems Company, Inc.
500 West Cummings Park, Suite 3000
Woburn, MA 01801

AFOSR STTR Phase I
Contract No. F49620-01-C-0055

Intelligent Record Linkage Techniques Based on Information Retrieval, Natural Language Processing, and Machine Learning

Final Report

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

Submitted to: Dr. Robert L. Herklotz, Technical Monitor

Prepared by: Eliot S.-M. Li (PI SSCI)
Charles Nicholas (PI UMBC)
Tim Oates (Co-PI UMBC)
Raman K. Mehra (PM)

Abstract

The objective of this STTR project is to develop an information management system to rapidly and accurately linking records of related information from web-based information sources. The sheer magnitude of information available online via the Internet has overwhelmed the ability of existing search tools to produce useful query responses. Current web-search techniques typically fail to correlate relevant documents that are identified in different ways, such as synonyms and acronyms (aliases). The challenge is to find an approach that can obtain highly accurate matches even when those documents do not share any obvious attributes with the query, and with minimal information requirement from the user. Latent Semantic Analysis (LSA) is a technique for identifying both semantically similar words and semantically similar documents. On the face of it, LSA should work well for the task of discovering aliases. That is, for a given word we can use LSA to produce a rank-ordered list of words that are semantically similar to it and aliases for the name should be high in this list. In this Phase I, we tested this conjecture empirically and found, surprisingly, that under a broad range of circumstances a straightforward application of LSA fails to rank true aliases highly. We then developed a two-stage algorithm that takes the output of LSA, creates a new set of pseudo-documents, and runs LSA again on these new documents. Empirical results show that this two-stage algorithm performs remarkably well in identifying aliases, even in those cases for which a single application of LSA fails miserably. University of Maryland (Baltimore County) is the research institute partner for this effort, under the direction of Professor Charles Nicholas and Tim Oates.

Acknowledgement

The authors would like to thank our technical monitor Dr. Robert L. Herklotz at AFOSR for his valuable assistance and comments regarding the project.

Contents

1 Introduction 7

2 Methodology 12

2.1 Latent Semantic Analysis 12

2.2 Empirical Evaluation of LSA for Finding Aliases 14

2.3 A Two-Stage Algorithm Based on LSA 18

2.4 Conclusions 20

3 Software User Manual 23

3.1 Installation 23

3.2 Performing Latent Semantic Analysis (LSA) 24

3.3 Finding Alises Using The Two-Stage LSA Algorithm 25

3.4 Tracking the Improvement 27

4 Future Work 29

4.1 Further development of record linkage techniques 29

4.1.1 Refine LSI techniques for record linkage 29

4.1.2 Improve record linkage performance using fusion techniques 30

4.1.3 Develop methods to learn symbolic rules for the identification of related terms and their relationship 32

4.2 Develop methods to generate metadata for terms 32

4.3 Develop methods for active exploration of open source content 34

List of Figures

2.1 Rank of `alqaeda2` as a function of t for various values of k under standard LSA. 16

2.2 Rank of `alqaeda2` as a function of t for various values of k under the two-stage LSA algorithm. 20

2.3 Rank of `alqaeda2` as a function of t for various values of k under the two-stage LSA algorithm (same vertical scale as Figure 2.1). 21

Chapter 1

Introduction

One of the main goals of DoD's Joint Vision 2020 is full spectrum dominance based on information superiority. It is expected that future combat planning and execution will be much faster, involve smaller forces that are highly autonomous. To achieve these goals, warfighters and commanders at all levels need to be able to control their assets and apply them with high degree of precision and confidence, based on comprehensive situational awareness of the environment and of enemy actions and intentions. This translates to increased requirements for timely intelligence, greater situational awareness, and an integrated operational picture of the battlefield.

Even though a lot of information is readily available now due to the rapid increase in surveillance assets, computer processing power, and storage space, it does not automatically give commanders information superiority over the adversaries. Even with today's technology, major obstaclesremain in achieving this goal, including:

- **Information overload:** Commanders can easily be overwhelmed by the sheer quantity of information that is available to them. Furthermore, information integrity can be difficult to verify in many cases, thus undermining the usefulness of available information.
- **Lack of interoperability:** Information obtained from different sources usually does not conform to the same standard or format, even when they are about the same entity or object. This makes integration or cross-checking of data a time-consuming process.
- **Immaturity in fusion:** Aggregating data from different sources to generate a reliable infor-

mation product requires more sophisticated data representation and organization, as well as reasoning or inferencing technologies, compared to what is available today.

- **Limits in display technology:** To get the maximum out of processed information, highly intuitive visualization and display technology is required to improve the cognitive capacity of users.

Although the automation of numerical data processing such as radar data or terrain map is common and has many successful applications, the problem is much more difficult for text data as it is much less amenable to representation in a form that enables computer processing. However, text information is simply too important to ignore. Commanders rely on intelligence in the form of news stories, manuals and reports to make strategic planning and battlefield decisions. The Internet also makes large amount of text data available through a mouseclick. If we do not take advantage of such data, our enemies will. After September 11, it was found that al Qaeda members obtained most of their intelligence from the public domain. One can argue that the battle for text information dominance is one that we have to win to survive.

The Joint Battlespace Infosphere (JBI) was proposed for enabling the US to achieve information dominance in both text and non-text information. With respect to text information, the JBI is required to provide the following capabilities in order to support intelligence gathering and analysis, and ultimately strategic decision making:

- Collect relevant documents from various sources.
- Verify information integrity.
- Organize the text information.
- Reason through text information.
- Fuse the information to produce high-level knowledge.
- Distribute information in a context sensitive manner.

Semantic web technologies, including XML (eXtended Markup Language), RDF (Resource Description Framework), and DAML (DARPA Agent Markup Language), are widely expected to be the building blocks of the JBI, as they are supposed to allow computers to understand webpages through the metadata, or tags that markup the terms, in them. However, merely defining the standard for marking up documents is not enough. The potential of such semantic web technologies relies critically on efficient and reliable ways of automatically inputting metadata on (tagging) text documents, i.e. without human intervention. This implies that unless there is an intelligent method of building metadata database and assigning them to terms in text documents, the usefulness of such semantic web technologies will be very limited, and unlikely to realize the JBI vision.

Based on the discussion thus far, it is obvious that significant advances in the following technologies are needed for successful implementation and deployment of the JBI:

- Text information retrieval with high precision and recall rate.
- Context-sensitive representation of text information and relationships between documents.
- Automatic generation of metadata for large corpora.
- Ontology construction.
- Information fusion for text data at level 2-4 [21].

These will also be the enabling technologies for automatic *deception detection*, a problem of growing importance due to the elusive nature of the war against terrorism. Well-defined logical representation of relationship between text records and terms will allow machine to reason through the records to understand the claims, and identify critical information that is available in the open source for verifying the claim.

In this Phase I project, we demonstrated the feasibility of using Latent Semantic Analysis (LSA) to identify synonyms or terms that are similar in meaning or concept within a text corpus. The approach holds great promise in overcoming a major obstacle in information assurance, i.e. building a comprehensive and context-sensitive ontology. Specifically, we considered the following problem: Even though two documents on the web may be on precisely the same topic, they may

use different terminology. Different terms that refer to the same entity may be synonyms (e.g. car vs. automobile), spelling variations (e.g. al Qaeda vs. al Qaida), abbreviations (e.g. People's Republic of China vs. PRC), or valid ways of naming the same entity (e.g. Bombay vs. Mumbai). When an entity has multiple names, searching for documents about that entity using only one of its names as a query term may cause relevant documents to be missed.

Our goal for Phase I was to develop a system that helps human users discover alternative names, or *aliases*, for an entity given just one of its names. Given a collection of documents and a name, the system should return a rank-ordered list of candidate aliases such that true aliases are high in the list and thus can be easily selected by the user. Such a system leverages both the strengths of machines in sifting through large amounts of information quickly and the strengths of humans in using background knowledge to reason about which candidates are in fact true aliases.

Latent Semantic Analysis (LSA) is a technique for identifying both semantically similar words and semantically similar documents [8]. It was developed to address problems with early information retrieval systems that performed exact word matching. That is, if you submitted the query `car` against a document collection, you would get all and only documents containing `car`. You would get no documents containing `automobile` unless they also contained `car`. LSA considers the words that co-occur in documents with `car` and the words that co-occur in documents with `automobile`, and given that they overlap significantly is able to determine that `car` and `automobile` are semantically related and that documents containing either word should be returned for the query `car`.

On the face of it, LSA should work well for the task of discovering aliases. That is, given a name we can use LSA to produce a rank-ordered list of words that are semantically similar to it and aliases for the name should be high in this list. We tested this conjecture empirically and found, surprisingly, that under a broad range of circumstances a straightforward application of LSA fails to rank true aliases highly. We then developed a two-stage algorithm that takes the output of LSA, creates a new set of pseudo-documents, and runs LSA again on these new documents. Empirical results show that the two-stage algorithm performs remarkably well, even in those cases for which a single application of LSA fails miserably. This methodology will be described in detail in chapter 2.

As part of the deliverables under this project, we have developed a software tool for finding the aliases of a given word in a corpus. The software tool is written in java and C, and currently runs on Unix platforms only due to the fact that it uses a package to perform Singular Value Decomposition (SVD) that compiles on Unix only. A detail user manual for the software tool in included in chapter 3 of this report.

In future, we plan to continue to develop techniques for efficient discovery of relevant information from large corpora and organize them in a form that can support human and machine analysis. These techniques will be essential for the advancement of the JBI enabling technologies listed above. The ultimate objective is to develop an Intelligent Information Management System (IIMS) that can support the envisioned JBI capabilities. To achieve the above objective, we plan to continue to improve record linkage performance of our algorithms by (i) refining the LSA method to incorporate recent results; (ii) combining results from multiple algorithms using fusion techniques; and (iii) developing methods to learn symbolic rules to complement the LSI methods for record linkage. We also plan to investigate and develop machine learning-based methods to generate metadata for text documents based on sample tagged-documents and linkages discovered between them. Such technology is crucial for the construction of the JBI ontology, which in turn will support the JBI's Structured Common Representation (SCR). Finally, we plan to develop methods for active exploration of open source content to retrieve additional information for the purpose of claim verification or deception detection. We plan to integrate all the algorithms developed into an object-oriented information system to provide a wide range of intelligence support to commanders and analysts. These plans will be discussed in more detail in chapter 4.

Chapter 2

Methodology

In this chapter we give details on the methodology that we adopted in this Phase I effort. We first review latent semantic analysis and describe the results of the experiments aimed at evaluating the utility of LSA in finding aliases. Then we describe the two-stage algorithm we developed, the motivation behind it, and compares its performance to that of standard LSA. Finally, we give a few concluding remarks that summarize the lessons learnt during this project.

2.1 Latent Semantic Analysis

The first step in information retrieval is to represent the document corpus under consideration as a term-document matrix in which the rows of the matrix correspond to words (or terms), the columns to documents, and the cells to the number of times a given word occurs in a given document. Specifically, given n documents that collectively contain m unique words, term-document matrix A is an $m \times n$ matrix such that $a_{i,j}$ is the number of times that word i occurs in document j. Note that although the dimension of the matrix A is huge, it is typically extremely sparse, with over 95% of the entries being zero.

Intuitively, if two columns of A are similar, then the corresponding documents contain similar words and are therefore likely to be semantically related. Matrix A defines an m-dimensional space in which each document corresponds to a point and similarity between documents can be measured according to some metric between points in this space. Likewise, if two rows of A are similar, then

the corresponding words occur in most of the same documents and are likely to be semantically related. In this case, A defines an n-dimensional space in which each word corresponds to a point and similarity between words can be measured by distance between points.

Unfortunately, this approach breaks down for two reasons. First, two words may be semantically identical yet never co-occur in the same document. For example, one author might always use the word `automobile` and another the word `car`. Second, even if two documents are about cars and use the word `car` rather than `automobile`, these documents may contain a great many other words that are peripheral to the topic and not shared. When comparing document vectors, the negative effects of these words can overwhelm the positive effects of words related to the topic which may be fewer in number though more semantically relevant.

Latent Semantic Analysis (LSA) overcomes these problems by projecting the row and column vectors of A into a lower dimensional space, one in which comparisons of these vectors are less susceptible to the effects of variance in word selection. This is accomplished by first computing the singular value decomposition (SVD) of A. The SVD of an $m \times n$ matrix A is three matrices – an $m \times n$ orthogonal matrix U, an $n \times n$ diagonal matrix Σ, and an $n \times n$ orthogonal matrix T – that can be multiplied together to recover A as follows:

$$A = U\Sigma V^T$$

The elements along the diagonal of Σ are the *singular values* of A and are denoted σ_i.

After computing the SVD of A, the singular values are sorted in non-increasing order so that $\sigma_i \geq \sigma_j$ for $i < j$. Note that re-ordering the elements on the diagonal of Σ will require re-ordering the columns of U and the rows of V^T to maintain the property that $A = U\Sigma V^T$. The projection of the row and column vectors of A into a lower dimensional space is accomplished by setting all but the k largest singular values to zero, yielding Σ_k. This effectively truncates U to a $n \times k$ matrix and V^T to be a $k \times n$ matrix because when computing $U\Sigma_k V^T$ only the first k columns of U and the first k rows of V^T enter into the computation. Note that, in general, $A \neq A_k$. However, A_k is the best approximation to A of any rank-k matrix in the least squares sense [12].

The truncated SVD of A yields an embedding of words and documents in a k-dimensional space where, typically, $k \ll n$ as A is extremely sparse. Just as each row of A corresponds to a

word and each column of A corresponds to a document, each row of U corresponds to a word and each column of V^T corresponds to a document. The similarity of two words in the k-dimensional space is determined by comparing the first k elements of the corresponding rows of the U matrix. Likewise, the similarity of two documents in this new space is determined by comparing the first k elements of the corresponding columns of V^T. Given two points in an appropriate k-dimensional space (i.e. two words or two documents), commonly used similarity metrics include Euclidean distance between the points and the cosine of the angle between the vectors from the origin of that space to the points.

For a given term-document matrix, A, the value of k has significant impact on the perceived similarity of words and documents. When $k = n$ the results are exactly the same as those obtained when using A. As k is reduced the words and documents are projected into spaces of increasingly small dimension. At "good" values of k semantically similar words will cluster together, as will semantically similar documents, but if k is too small multiple clusters will be projected onto one another and words/documents that are semantically unrelated will be close together erroneously.

2.2 Empirical Evaluation of LSA for Finding Aliases

Because LSA can and has been used (with much success) to identify semantically related words in document collections [2], it seems reasonable to expect that it would be a valuable tool in the search for aliases. If two different words are in fact alternative names for precisely the same entity, e.g. `al Qaeda` and `al Qaida`, those two words are more than semantically related, they are semantically identical. Given a document collection and a name one can build a term-document matrix, compute its SVD, retain the k largest singular values, and determine the semantic similarity of all words in the collection to the given name in the resulting k-dimensional space. A human user can then be presented with a list of words ordered by their similarity to the name. Ideally, true aliases for the name will be high in this list.

Given an algorithm that takes as input a name and a document collection and produces as output a rank-ordered list of candidate aliases for the name, our general approach to evaluating the algorithm is as follows. Let N and D be the name and document collection, respectively. Let S_1

and S_2 be two arbitrary strings that do not occur in D, and let p be a percentage ($0 \leq p \leq 100$). We identify all documents in D that contain the string N and replace N with S_1 in $p\%$ of these documents and with S_2 in the others. That is, we artificially create two names, S_1 and S_2, for the entity whose original name in D is N. The performance of the algorithm is then measured by the rank of S_2 in the list of candidate aliases for S_1.

The experiments reported in this section empirically evaluate the utility of LSA with respect to the alias identification problem on a corpus of 77 documents taken from `www.cnn.com`. The documents contain 516 words on average, with the smallest containing 131 words and the largest 1923 words. The term `al Qaeda` occurs one or more times in 49 of the documents, though most of the documents that mention al Qaeda are not primarily about that organization. For example, one of these documents is about the effects of the events of September 11th on US immigration policies, another is about a wrongful death lawsuit filed against airlines whose planes were involved, and several others are about individuals who are suspected of being members of al Qaeda. The remaining 28 documents are on a variety of topics such as politics, sports, and entertainment.

The experiments reported below varied two parameters - k and t - and used the single value $p = 50$. The first and third of these parameters have already been introduced - k is the dimensionality of the space returned by LSA and p is the percentage of documents in which name N is replaced with string S_1. In this case, $N =$ `al Qaeda`, $S_1 =$ `alqaeda1`, and $S_2 =$ `alqaeda2`. That is, LSA was used to find candidate aliases for `alqaeda1` with `alqaeda2` being the only true alias. Because $p = 50$, half of the documents were randomly selected for replacement of N by S_1 with N being replaced by S_2 is the other half. The final parameter, t, is a threshold used to filter words based on the logarithm of their TF/IDF scores as computed by the WONDIR search engine, which was used for all document management tasks [6]. Filtering on TF/IDF scores, which occurs before building the term-document matrix, is a common practice in information retrieval [19]. The TF/IDF score is a measure of "importance" of each word in the corpus based on its frequency of occurrence and its uniqueness. Larger values correspond to more aggressive filtering with the goal of removing words that are irrelevant and thus might distract the algorithm.

When using LSA to find aliases for a name, it is not feasible for the user to experiment with many different values of k and t because each pair of values will produce a rank-ordered list that

must be manually scanned for aliases. In selecting t there is a tension between choosing high values to remove as many extraneous words as possible, and choosing low values to ensure that no potential aliases are filtered out. Clearly, it is important for an alias discovery algorithm to be robust with respect to the choice of parameters.

Figure 2.1 shows a plot of the rank of `alqaeda2` as a function of t in the list of candidate aliases for `alqaeda1` returned by LSA. There is one such curve for each of four different values of k. Smaller ranks are better because they correspond to `alqaeda2` being higher in the list presented to the user. Note the vertical scale, which has ranks ranging from a maximum of 246 to a minimum of 5. That is, in at least one case the user would have to scan through 245 words before seeing a true alias for `alqaeda1` (i.e. when $k = 5$ and $t = 1.7$).

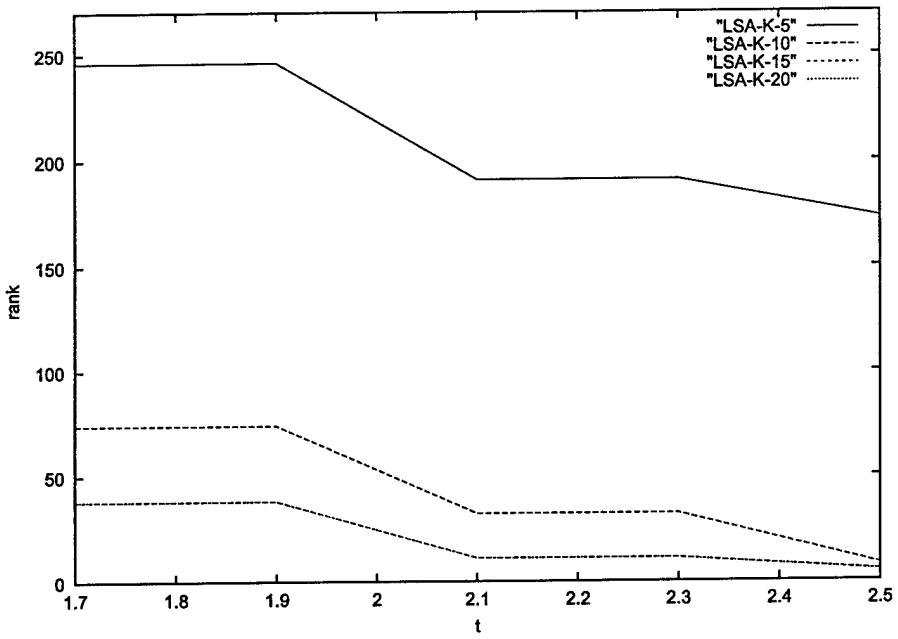

Figure 2.1: Rank of `alqaeda2` as a function of t for various values of k under standard LSA.

There are two notable trends in the data. First, ranks generally improve with increasing k. The one exception is for $k = 15$ and $k = 20$ which produce the same ranks for all values of t. Apparently, 15 or more singular values are sufficient for accounting for the vast majority of the variance in the term-document matrix, and including additional singular values has no effect on relative ranks and little effect on the corresponding similarity scores. Second, for a given value of

k, larger values of t produce better ranks. That is, all of the curves slope downward from left to right.

Table 2.1 shows the top 10 candidate aliases for $t = 2.5$ and three different values of k (there is no need to show all four as the lists for $k = 15$ and $k = 20$ are the same). The true alias `alqaeda2` appears in the lists for $k = 10$ and $k = 20$ but not in the list for $k = 5$. In general, the lists contain words that are indeed semantically related to `alqaeda1`, including names of individuals who are thought to be members of al Qaeda. Clearly, LSA is behaving as expected with respect to the discovery of semantically related words, yet there are large ranges of parameter values for which the results are poor from the standpoint of alias discovery.

$k = 5$	$k = 10$	$k = 20$
arrested	zubaydah	zubaydah
government	raids	ressam
ressam	ressam	raids
lindh	pakistani	hamdi
zubaydah	hamdi	alqaeda2
raids	soldier	pakistani
attacks	trial	trial
brahim	alqaeda2	soldier
passengers	pakistan	pakistan
virginia	walker	lindh

Table 2.1: The top 10 candidate aliases for `alqaeda1` for $t = 2.5$ and various values of k.

Empirically, it appears that one should choose both k and t to be large. The danger is that with large t true aliases could be filtered out, and with large k one might simply reconstruct the term-document matrix and lose the beneficial effects of SVD with respect to dealing with variability in word choice. In addition, there is no way for a user running LSA on a new document collection to determine "good" values of k and t a priori. What is needed is an algorithm that works well over a wide range of values of these parameters. Such an algorithm is described in the next section.

2.3 A Two-Stage Algorithm Based on LSA

It is instructive to look at the words in table 2.1 that are not aliases. It is clear that almost all of them refer to entities that are not the same kind of thing as the entity known as al Qaeda. That is, al Qaeda is an organization, whereas many of the other words in the table refer to people (e.g. Lindh, Zubaydah, soldier, passengers), places (e.g. Virginia, Pakistan), and activities (e.g. raids, attacks). If one could determine that al Qaeda is an organization and that Lindh is a person then it would be possible to remove Lindh from the list of candidate aliases.

Making ontological distinctions of the type just described requires knowledge about the entities involved. If we had this knowledge there would be no alias discovery problem, so we cannot assume that it exists. Rather, the second stage of our algorithm attempts to determine the degree of similarity between the ontological status of two entities. This is accomplished by looking at the words that surround occurrences of candidate aliases. For example, sentences that mention al Qaeda in our document collection contain phrases such as the following: "list of al Qaeda leaders"; "most senior al Qaeda member captured"; "alleged al Qaeda operatives". Note that the words near occurrences of al Qaeda are indicative of the fact that it is an organization with "leaders", "members", and "operatives". In contrast, the words in phrases surrounding occurrences of Lindh are suggestive of the fact that the entity with that name is a person: "photograph showing Lindh blindfolded", "compared with Lindh, the 21-year-old American"; "Lindh pleaded not guilty".

Although it is unreasonable to have a human user scan through a list of a few hundred candidate aliases, it is not unreasonable for a computer program to take the same list and reorder it based on the degree of ontological similarity between the candidates and the name used to generate the list. Our two-stage algorithm implements this ideas as follows: Given a list L of the top several hundred aliases for N as determined by LSA, create a new document collection, D', containing one document for each element of L. For a given word in L, the corresponding document in D' contains the text in a small window around every occurrence of the word in each document in the original document collection. LSA is then run on D' and a rank ordered list of candidate aliases for N is created as before.

The second phase of the algorithm, which is simply a second run of LSA, attempts to determine

ontological similarity by considering just the local context around occurrences of words rather than the global (i.e. entire document) context considered in the first application of LSA. For example, the document in D' corresponding to the term `al Qaeda` might contain the following text:

```
list of al Qaeda leaders
most senior al Qaeda member captured
alleged al Qaeda operatives
```

We have not modified LSA to focus on words that occur in close proximity to candidate aliases. Rather, we have created a new set of documents such that each document contains all of the local context for a single word.

In the experiments reported below, a document was created in D' for each of the top 250 words in the rank-ordered list returned by the first run of LSA. The local context around each occurrence of a word was identified by growing a window around the word both backward from its beginning and forward from its end by 10 characters. Any words that wholly or partially fell within this window were considered local context. The goal was to get one or two words before and after each occurrence of each candidate alias.

The results of applying the two-stage algorithm to the document collection and task described in the previous section are shown in figures 2.2 and 2.3 which present the same information as figure 2.1. Note, however, that the vertical scale of figure 2.2 is very different from that of figure 2.1. The maximum rank is 6 and the minimum rank is 2. That is, regardless of the values of t and k, the two-stage algorithm always returned `alqaeda2` with a rank no greater than 6. This is a dramatic improvement, especially in comparison to runs of LSA with small values of k and t for which `alqaeda2` could literally be a couple of hundred entries down from the top.

Another prominent difference between figures 2.1 and 2.2 is that there are no trends in the latter. That is, the performance of the two-stage algorithm in uniformly good, regardless of the values of k and t. Figure 2.3 shows the same information as figure 2.2 except the vertical scale is the same as figure 2.1. This makes it possible to visually compare the relative performance of LSA and the two-stage algorithm. There is no parameter combination for which the performance

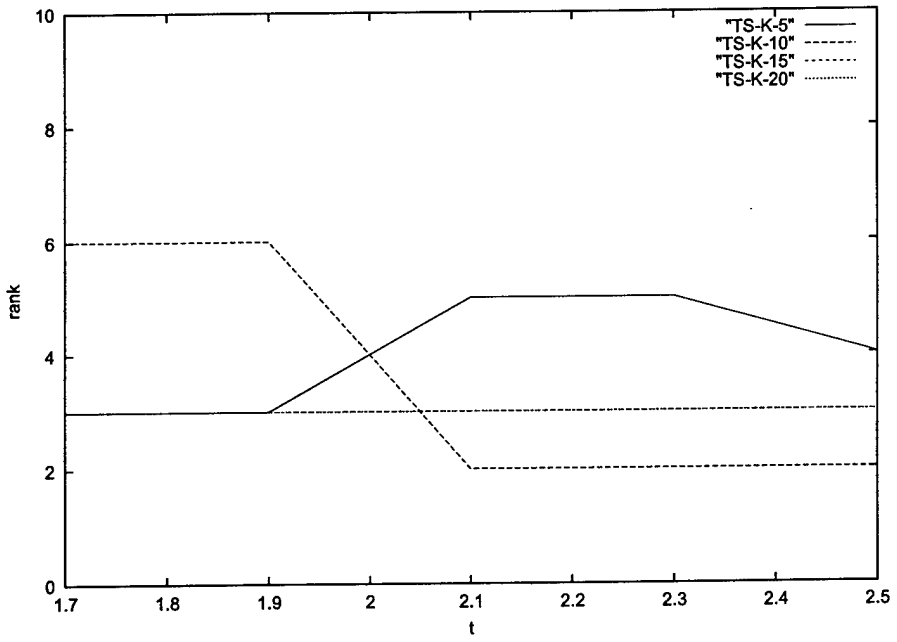

Figure 2.2: Rank of `alqaeda2` as a function of t for various values of k under the two-stage LSA algorithm.

of LSA is equal to or better than that of the two-stage algorithm.

Table 2.2 shows the top 10 candidate aliases for $t = 2.5$ and three different values of k (again, there is no need to show all four as the lists for $k = 15$ and $k = 20$ are the same because the output from the first run of LSA is the same in those cases). Note that the words in this list are qualitatively rather different than those in table 2.1. There are many words related to organizations (e.g. cell, terrorism, operation, network) and only a single person name (i.e. the last name of CIA director George Tenet).

2.4 Conclusions

Two conclusions can be arrived at based on this Phase I investigation. First, it demonstrates empirically that the success of a straightforward application of LSA to the task of alias discovery is highly dependent on choosing good parameters. While there are some sets of parameter values for which LSA performs well, there are many others for which it performs poorly. Second, the paper describes a two-stage algorithm for alias discovery that is based on LSA and performs significantly

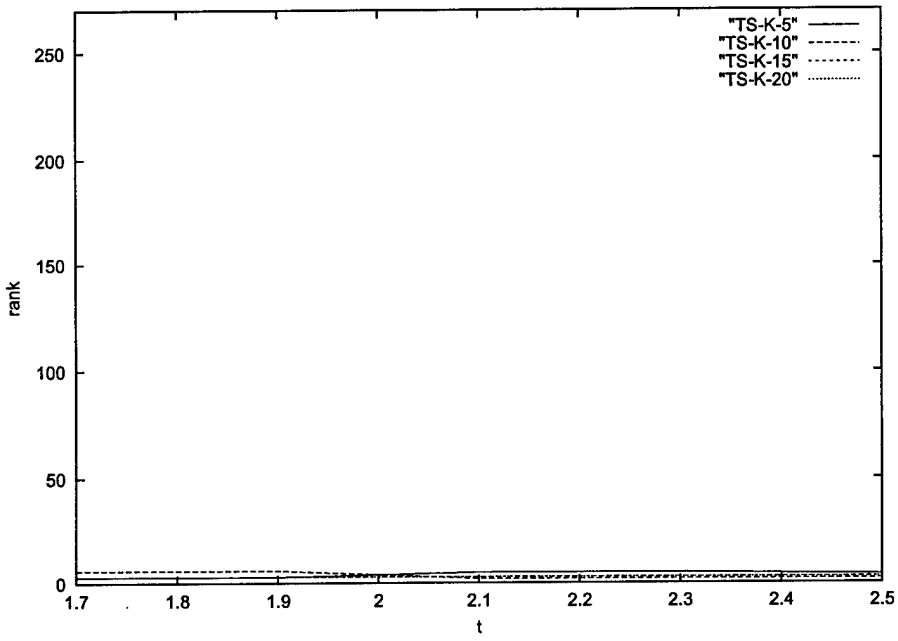

Figure 2.3: Rank of `alqaeda2` as a function of t for various values of k under the two-stage LSA algorithm (same vertical scale as Figure 2.1).

better. The second stage of the algorithm re-orders the output of LSA so that candidate aliases with similar ontological status as the target name are ranked highly. This application of LSA is similar in spirit to explicit clustering of words based on similarity of local context which has been found to yield groupings of words based on ontological similarity [3].

$k = 5$	$k = 10$	$k = 20$
tenet	cells	cells
suspected	alqaeda2	network
warned	network	alqaeda2
alqaeda2	suspects	cell
terrorism	germany	terrorist
terrorist	laden	alleged
anaconda	alleged	suspects
potential	cell	laden
operation	terrorist	singapore
operations	warned	germany

Table 2.2: The top 10 candidate aliases for `alqaeda1` for $t = 2.5$ and various values of k.

Chapter 3

Software User Manual

As part of the deliverables in this project, we have developed a software package for identifying aliases of any chosen word in a given corpus. This chapter will describe the use of this software package in detail.

3.1 Installation

All the files in the software package are stored in a zipped format (tarred and gzipped). In order to unzip the file, do the following on command prompt:

```
tar -xvzf SSC_STTR.tar.gz
```

This will unzip the files and create a directory called `UMBC_SSC`

This directory contain the source code of the algorithm written in java and C. The documents that have to be tested are placed in the subdirectory `documents`. Currently, this directory contains 77 documents in plain text format. The term*al Qaeda* occurs one or more times in 49 of these documents. The remaining 28 documents are on a variety of topics such as politics, sports, and entertainment. Under the `documents` directory, there is a subdirectory called `window_docs`. It is created to hold auto-generated documents during the second stage of the two-stage LSA to be described in section 3.3. You can remove, add, or replace any or all the text documents in the subdirectory `documents`. However, do not delete the subdirectory `window_docs` at any time.

To compile the Java files, simply type the following in the UMBC_SSC directory

```
javac *.java
```

This will generate the executable `.class` files.

NOTE: We have used a public domain package SVDPACK for performing singular value decomposition of the word-document matrices in order to save development time. As a result, this tool can only be compiled and used on sunservers.

3.2 Performing Latent Semantic Analysis (LSA)

To use this tool to perform standard LSA for the purpose of finding alias of a particular term *term1* within a document corpus, follow the steps below:

1. Remove all existing files in the directory `documents`, except the subdirectory `window_docs`, and put the testing corpus in the directory. Note that the only valid format for documents in the corpus is plain text.
2. Choose parameters TFIDF threshold t and desired reduced dimension after SVD k.
3. Type the following at the command prompt

   `java RelatedWords` *term1* `-t` t `-k` k
4. The result is written in the text file `ammaF.`t`.k`k`.30.10.txt`.
5. To save this file for future analysis, move it to the subdirectory `results`.

In the file `ammaF.`t`.k`k`.30.10.txt`, each row contains *term1*, another word in the corpus that is considered as important based on their TFIDF score, and a number. Words in the corpus is rank ordered by how likely it is an alias of *term1* in this file. In other words, the higher a word is listed in this file, the more likely it is an alias of *term1*, or closely related to it. The number at the end of each row in this file can be considered as the score of how close the word in that row to *term1*.

3.3 Finding Alises Using The Two-Stage LSA Algorithm

Due to the poor performance of standard LSA for finding aliases, we have developed a two-stage LSA algorithm to improve the performance. This algorithm is essentially two consecutive run of the standard LSA, with the second run performed on a set of artificially generated documents that contains words in the vicinity of the words that are considered potential aliases of *term1* in the first run. Those artificially generated documents are stored in the directory `documents/window_docs`.

We have written the following script `scriptTwoPhase.sh` to simplify the task of running the two-stage LSA algorithm:

```
#!/bin/sh
for t in t
do
    for k in k
    do
       for w in w
       do
          for ws in ws
          do
             java RelatedWords term1 -t t -k k -w w -ws ws
             java DeleteAll
             java test ws
             java RelatedWords term1 -t t -k k -w w -ws ws WINDOW
          done
       done
    done
done
```

Variables in italics have to be entered first before running the algorithm.

Running the two-stage LSA algorithm

To find the alias of a chosen word *term1* using the two-stage LSA algorithm, follow the procedure below:

1. Remove all existing files in the directory `documents`, except the subdirectory `window_docs`, and put the testing corpus in the directory. Note that the only valid format for documents in the corpus is plain text.

2. Choose parameters TFIDF threshold t, desired reduced dimension after SVD k, number of important words that will be required in the second stage w, and the window size ws.

3. Edit the script file `scriptTwoPhase.sh` to make the program run for the chosen parameter values.

4. Run the script `scriptTwoPhase.sh` by typing the following at the command prompt

   `scriptTwoPhase`

   OR

   `./scriptTwoPhase.sh`

5. The result is written in the text files

   `ammaF.`t`.k`k`.`ws`.`w`.txt` and

   `ammaS.`t`.k`k`.`ws`.`w`.txt`.

   The first file contains the result of the first stage and the second file contains the result of the second stage.

6. Rerunning the script will overwrite existing amma* files with the same name. To save these files for future analysis, move it to the subdirectory `results`.

The format of the output files is the same as the standard LSA described in section 3.2. In general, the ranking in the second stage is significantly better than that in the first stage. The next section describes an utility that we developed to make tracking the improvement from the first to the second stage much easier.

3.4 Tracking the Improvement

Follow these steps to find the rank of any particular word *term2* in the corpus with respect to the word *term1* at the end of each stage:

1. Move all the amma* files into the directory `Statistics/vbtest`. Notice that there are 2 amma files for each parameter setting (`ammaF*` and `ammaS*`), and both need to be moved to this directory.

2. Get into the `Statistics` directory. There is a java file called `Statistics1.java` in there. Compile it by:

   ```
   javac Statistics1.java
   ```

3. Obtain the ranking of *term2* at the end of each stage by typing the following at the command prompt

   ```
   java Statistics1 term2 term1
   ```

The result of this utility is written to a text file `FINAL.txt`. Below is part of a sample `FINAL.txt` file.

```
1.7.k10.10.250.txt      74 / 541      7 / 424
1.7.k15.10.250.txt      38 / 542      4 / 500
1.7.k20.10.250.txt      38 / 542      4 / 500
1.7.k5.10.250.txt       246 / 544     4 / 478
1.9.k10.10.250.txt      74 / 541      7 / 424
1.9.k15.10.250.txt      38 / 542      4 / 500
1.9.k20.10.250.txt      38 / 542      4 / 500
1.9.k5.10.250.txt       246 / 544     4 / 478
2.1.k10.10.250.txt      32 / 302      3 / 343
```

In this file, the first column is the part of the file name that identify the parameters used in the experiment. For instance, the first row consists of results of an experiment with parameter $t = 1.7$,

$k = 10$, $ws = 10$, and $w = 250$. The second column is the rank/(total number of words) after first stage The third column is the rank/(total number of words) after second stage.

This output file provides a convenient way to verify our hypothesis, and allows the user to compare the relative merits of choosing different parameter values easily.

Chapter 4

Future Work

Based on the foundation laid by our Phase I work, our next objectives include develop techniques for efficient discovery of relevant linkages among text records, use the discovered linkages to organize the records, and perform active retrieval of new information, to support human and machine reasoning. We plan to implement the techniques and algorithms in an integrated Intelligent Information Management System (IIMS) prototype. The resulting system is expected to greatly enhance the military capability in intelligence gathering, as well as information fusion, situation awareness, and deception detection. The following sections describe areas where further research and development are needed in order to achieve the above objectives.

4.1 Further development of record linkage techniques

Further improvement of the record linkage performance of the algorithm developed in Phase I is needed to provide better recall and precision rates of information retrieval. We have identified three different approaches to accomplish this goal, as described in the following subsections.

4.1.1 Refine LSI techniques for record linkage

Our Phase I work is based on Latent Semantic Indexing (also known as Latent Semantic Analysis). LSI was developed at Bellcore during the late 1980's and early 1990's [8]. LSI is one of several techniques for analyzing a collection of documents in which the problem of dimension reduction

is addressed. The basic idea is as follows: Each document in a corpus is represented as a vector in a high dimensional vector space. The documents form the columns of a matrix, and each term that appears in the corpus, in any document, corresponds to a row of that same matrix. Each entry in the matrix indicates the number of times the term corresponding to that row appears in the document corresponding to that column. The goal of LSI is to decompose this matrix in a way that makes its structure more apparent, while reducing noise.

LSI can be considered as an enhancement to the familiar vector-space model of information retrieval. Typically, authors use many words to describe the same idea or fact, and those words will appear in only a few contexts. LSI attempts to highlight these patterns of how words are used within a document collection. By grouping together the word co-occurrence patterns that characterize groups of documents, the "latent semantics" of the collection terms is described. Themes in the document collection arise from subsets of documents with similar word co-occurrences. The strength of LSI is that it can identify related documents by virtue of their references to the same concepts, as opposed to simply using identical words and phrases. In other words, LSI was intended to identify documents that referred to the same concepts. In some previous work, LSI was used to determine if documents are related to the same topics. LSI was also used to study certain properties of documents, such as whether the documents in question are written by the same author.

During this Phase I project we developed a two-stage LSI algorithm for synonym identification and demonstrated significant improvement compared to standard LSI. In future we plan to incorporate recent developments in LSI that hold great potential in improving record linkage performance. In particular, the iterative document vector scaling technique proposed by Ando [1] will likely improve the precision of our record linkage algorithm. Another enhancement we plan to consider is to incorporate user ratings of documents, which can be obtained by web usage mining [5].

4.1.2 Improve record linkage performance using fusion techniques

The integration of multiple models for solving a *common* problem often leads to a composite global model that provides more accurate and reliable estimates or decisions. A strong motivation

for such systems was voiced by Kanal in his classic 1974 paper [14], "It is now recognized that the key to pattern recognition problems does not lie wholly in learning machines, statistical approaches, spatial, filtering,... No single model exists for all pattern recognition problems and no single technique is applicable to all problems. Rather what we have is a bag of tools and a bag of problems," and subsequently widely recognized in diverse communities including AI and machine learning.

Integration of multiple models for classification [20] and for multi-sensor data fusion [7] in particular have met with great success. Classifier combining provides statistically significant improvement in performance along with tighter confidence intervals [11]. Combining is an effective way of reducing model variance, and in certain situations it also reduces bias. It works best when each learner is well trained, but different learners generalize in different ways, i.e., there is diversity in the ensemble.

Another way of combining results is to apply divide-and-conquer approaches, where relatively simple learners specialize in different parts of the input-output space, and the total model is a (possibly soft) union of such simpler models. Such approaches include "mixtures-of-experts" (MOE), local linear regression, mixture models such as mixture of principal components, CART/MARS, adaptive subspace models, etc.[17]. Note that, in contrast to ensembles, the combination/selection of individual outputs now depends on the current input. Modular learning systems are based on the precept that learning a large number of simple local concepts is both easier and more useful than learning a single complex global concept. They are often found to learn concepts more effectively (better performance) and more efficiently (faster learning). Using simpler local models have several added advantages such as easier interpretability, better local tuning or adaptation, easier incorporation of prior knowledge, and less susceptibility to the curse of dimensionality.

A recent paper has shown the advantages of classifier fusion for text documents [15]. But there has been little exploration of combining methods for linking text documents, or for mixed (supervised/unsupervised) training. We have considerable in-house expertise of these fusion techniques with several application successes, and feel that they hold great potential for providing *substantial* improvements. In particular, the MOE approach can automatically partition the documents into different types which can be tackled more effectively. For example documents that can be linked

by identifying a key phrase can be separated by those needing a bag-of-words model, and custom models can be constructed for both groups. Both machine learning/AI inspired approaches (meta-learning) and Neural Networks (ensembles, MOE) will be considered for the linkage fusion studies.

4.1.3 Develop methods to learn symbolic rules for the identification of related terms and their relationship

LSI can be considered as a statistical technique that identify synonyms based on their frequency of occurrence and that of terms in close proximity with them. Given that LSI has identified, for example, "al Qaeda" and "al Qaida" as synonyms, we now have the opportunity to learn symbolic rules that have the potential to identify still other synonyms that occur infrequently enough that LSI cannot find them. More specifically, we can learn one set of symbolic rules that dictates how and where "al Qaeda" appear in documents, and another set for "al Qaida". Then we can use the intersection of these two rule sets to find other synonyms of "al Qaeda". This approach fits well with the idea that fusion should move from subsymbolic (LSI) to symbolic (rules) knowledge that can be used for reasoning.

4.2 Develop methods to generate metadata for terms

One crucial element of the JBI is its ontology. This ontology provides the military "semantics" or meaning to the elements of the JBI's Structured Common Representation (SCR). The SCR in turn supports various processes such as information visualization, customization of information for individual users, and drill down capability. So it is vital that the JBI ontology be capable of supporting the information needs of different users under different mission scenarios.

One of the key components of an ontology is its metadata. Metadata facilitate quicker and more accurate search - if the user is interested in the feeding habits of bats, he can look for terms with metadata "mammals" rather than with "sports". Some quick experiments on internet search with/without using such metadata within the webpage will convince the reader of this. They are also key to content management, updating and understanding.

Despite its importance, inputing metadata to text documents is a labor-intensive and time-consuming process, and thus a very limited portion of existing documents are tagged with meta-data. Under this task we will investigate the use of machine learning techniques to learn rules that can generate metadata for large corpora automatically.

Many of the techniques used in modern rule learning have been adapted from classic decision tree learning (eg. [22]). Most widely-used decision tree learning systems use an overfit-and-simplify learning strategy to handle noisy data: a hypothesis is formed by first growing a complex tree which overfits the data, and then pruning the complex tree. A variety of methods have been proposed to prune trees, but the most effective technique is called Reduced Error Pruning (REP). In REP for rules, the training data is first split into a growing set and a pruning set. First, an initial rule set is formed that overfits the growing set, using some heuristic method. This overlarge rule set is then repeatedly simplified by applying one of a set of pruning operators. At each stage of simplification, the pruning operator chosen is the one yielding the greatest reduction of error in the pruning set. Simplification ends when applying any pruning operator would increase the error on the pruning set. The REP scheme has been constantly refined, with various propositions for pruning rules, and reduction on the computational cost. One attractive REP based scheme is called RIPPER (Repeated Incremental Pruning to Produce Error Reduction), originally proposed in [23]. RIPPER was shown to outperform other rule based methods such as C4.5 and C4.5rules ([24]) on very noisy datasets, while scaling nearly linearly with the number of examples, in contrast to cubic scaling in C4.5 and C4.5rules. This improvement on computational cost permits the effective utilization of association rules on problems containing several hundreds of thousands of noisy sample data, such as would result from unstructured document streams.

Research topics under this area include:

1. For a given document to be tagged, what are the requirements on the training corpus (XML or RDF corpora) such that sufficient and relevant metadata can be learnt?
2. How to develop machine learning techniques that can learn rules to generate metadata efficiently?
3. Giventhe scarcity of training documents, is it possible to use the Expectation-Maximization

(EM) method to bootstrap a small number of XML or RDF documents with inferred supplemental information from documents without metadata?

4.3 Develop methods for active exploration of open source content

Information gathering and analysis is an iterative process. After information is gathered and filtered, the result of the analysis may found that additional information is required to substantiate the conclusion derived from the available information. A good example is deception detection: An adversary who deliberately posts deceptive information will never include sufficient information in the same source that enables human analysts to verify the claim. Additional research is always needed to gather sufficient information for this purpose. For instance, if John Doe is appointed as the president of an overseas exchange business with millions of dollars in revenues, then we expect his education level will be at least college graduate or higher. This will lead us to search for John Doe's resume or school record to verify if the appointment is under normal circumstances.

The problem of active exploration of new information has been extensively studied in the context of battlefield situation and is known as *sensor management* or *Level 4 data fusion*. In those cases the problem is how to orient or position sensors (radar, sonar, etc.) such that they will collect more data about the targets and improve the accuracy of their position or velocity estimates. In the case of intelligence data, the problem is much more complicated as that will involve reasoning through the collected data and figure out what additional information is needed, and where to find them.

Given the above characterization of what it means for information to be deceptive, what properties should an approach to detecting such information have? First, determining whether a given piece of information is improbable will typically depend on large amounts of background knowledge. In the example described above, this can include knowledge of the types of jobs a person might hold given their level of education or the types of contracts a business might win given their history and size. Therefore, approaches to detecting deceptive information should allow easy incorporation of background knowledge. Second, as in many other such endeavors, it is important

to avoid high false alarm rates while maintaining high true positive rates. Finally, approaches to detecting deceptive information should be as automated as possible without making human input to the process impossible.

As we will argue below, Bayesian Belief Networks (BBNs) [13, 9] can be used to efficiently operationalize our definition of deceptive information and they support all of the considerations given above. A BBN is a directed acyclic graph in which the nodes represent variables and the arcs represent probabilistic dependencies. For example, in the employment domain there might be one node that represents a person's level of education and another node that represents their current salary. Typically, the values taken by variables are discrete. So the education level might be `high-school-diploma`, `bachelor's-degree`, `master's-degree`, etc., and the salary might be 0K - 5K, 5K - 10K, 10K - 20K, etc. Because salary depends on education level, there would be an arc from the latter to the former.

BBNs support the goal of directing further information gathering activity to improve the confidence of any conclusion drawn from existing information. As described below, BBN can be used to identify deceptive information in a certain domain by asking how probable an item is given everything else that we know. The BBN will in general give us a probability distribution over values for that item. We can then use the BBN to reason about how that distribution will change if we gather specific additional information. In general, the fewer the number of variable values you know the less concentrated the mass of the distribution. That is, the less certain you are about the true value. As you gather additional information the mass of the distributions for the values that you don't know becomes more concentrated. The BBN can be used to reason about how the mass will change as you gather additional items of information, thereby focusing the search for information that will lead to more certain probability estimates and therefore improve confidence.

BBNs also support the detection of deceptive information though probabilistic inference. Given values for some subset of the nodes (i.e. variables) in a BBN, it is possible to compute probability distributions over the values of the other variables. For example, given the simple two-node BBN described above for the employment domain, if we know someone's educational level we can compute a probability distribution over salaries. (We can compute a probability distribution over salaries even if the educational level is unknown, but its mass will in general be more spread out.

There will be more on this shortly.) If we know that John Doe only completed high school and we learn that he currently has a salary of $150,000 per year, we can use the BBN to determine the probability of John Doe's salary given everything else that we know about him (i.e. his educational level). If we find that a salary of $150,000 per year is highly improbable given a high school education, then that piece of information is either inaccurate or deceptive. We might say that a piece of information is highly improbable if its probability is below some threshold value. BBNs offer an efficient and formally sound method for operationalizing our definition of what it means for information to be deceptive.

Finally, BBNs support the incorporation of background knowledge in at least two ways. First, it is possible for a human to specify the structure of the BBN, specifying what variables are important in a particular domain (i.e. the nodes) and how they are related to one another probabilistically (i.e. the arcs). Second, a human can specify the precise nature of the dependencies between variables in the form of Conditional Probability Tables (CPTs). Given the parents of a node in a BBN, the CPT for that node gives the probability of each value the node can have given every possible combination of values that the parents can have. The graphical nature of BBNs make it easy for domain experts to encode background knowledge in this way.

Although humans can specify both the structure and parameters (CPTs) of a BBN, there exist algorithms for learning both of these from data. That is, given a new domain it is possible to automatically learn a BBN that can be used to detect deceptive information. For example, given a database containing variables such as education level, current salary, employment history, and so on, it is possible to determine which variables should be connected via arcs and what the correspond CPTs are. Here again the human can enter the process by providing appropriate domain examples from which to learn the BBN, by partially specifying the structure or parameters of the BBN prior to learning, or by modifying the learned BBN.

In order to investigate BBNs as the representation and reasoning mechanism for deception detection agents, and direct the retrieval of additional information, the following steps have to be followed:

1. **Obtain test corpus:** Identify and obtain a database for some domain in which to test the

utility of BBNs. For research purposes, one possibility is to construct a database by gathering resumes from the Internet. Some fraction of them could be altered by hand to include information that is deceptive. Another possibility is to use the Securities and Exchange Commission's EDGAR system (`http://www.sec.gov/edgar.shtml`) to download filings for a set of similar companies and introduce deceptive information by hand.

2. **Determine relevant background knowledge:** Identify relevant background knowledge for the chosen domain(s) to support the next step.

3. **BBN Construction:** Hand-code a BBN, both the structure and the CPTs, for the chosen domain(s) and construct an algorithm for automatically identifying deceptive information in the modified database(s) using the BBN. This will involve treating each item of information for a person or company as unknown, using the BBN to compute a distribution over values for that item of information, and then determining the probability of the actual value in that distribution. If the probability is below some threshold the corresponding information will be identified as potentially deceptive.

4. **Examine BBN:** After tuning the BBN and the algorithm, one has to identify those features of the overall system that are most important for good performance using ablation studies.

5. **Develop methods for active search of critical information:** Analyse the BBN and develop algorithms that will identify the most critical item that is missing but will significantly change the confidence of the conclusion if it is available. This information will then be used to retrieve the information required to verify the conclusions.

Bibliography

[1] R. K. Ando. Latent semantic space: iterative scaling improves precision of inter-document similarity measurement. In *Proceedings of SIGIR 2000*, pages 216–223, Athens, Greece, Jul 2000. ACM.

[2] M. W. Berry and R. D Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numerical linear algebra with applications*, 3(4):301–327, 1996.

[3] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[4] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 181–188, 1991.

[5] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining World Wide Web browsing patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.

[6] R. S. Cost, S. Kallurkar, H. Majithia, C. Nicholas, and Y. Shi. Integrating distributed information sources with CARROT II. In *Proceedings of the Sixth International Workshop on Cooperative Information Agents*, 2002.

[7] B. Dasarathy. *Decision Fusion*. IEEE CS Press, Los Alamitos, CA, 1994.

[8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[10] E. A. Bier *et al.* . Toolglass and magic lenses: The see-through interface. In *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems*, volume 2 of Videos: Part II, pages 445–446, Boston, MA, 1994.

[11] J. Ghosh, S. Beck, and C.C. Chu. Evidence combination techniques for robust classification of short-duration oceanic signals. In *SPIE Conf. on Adaptive and Learning Systems, SPIE Proc. Vol. 1706*, pages 266–276, Orlando, Fl., April 1992.

[12] G. H. Golub and C. F. V Loan. *Matrix Computations*. Johns Hopkins University Press, 1989.

[13] D. Heckerman. Bayesian network for knowledge discovery. In U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, editors, *Advances in Knowledge Discovery and Data Mining*, pages 273–306. 1996.

[14] L. Kanal. Patterns in pattern recognition. *IEEE Trans. Information Theory*, IT-20:697–722, Nov. 1974.

[15] K-Y. Lai and W. Lam. Automatic textual document categorization using multiple similarity=based models. In *Proc. SDM 2001*, 2001.

[16] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[17] V. Ramamurti and J. Ghosh. Structurally adaptive modular networks for non-stationary environments. *IEEE-Transaction on Neural Networks*, 10(1):152–60, 1999.

[18] G. G. Robertson, S. K. Card, and J. D. Mackinlay. Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4):56–71, 1993.

[19] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[20] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[21] I.R. Goodman, R.P.S. Mahler, and H.T. Nguyen (1997) *Mathematics of Data Fusion*, Kluwer.

[22] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1993.

[23] W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[24] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1994.