On the Minimum Vocabulary Problem

N. Chandrasekharan
Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803

R. Sridhar
School of Library and Information Science, Louisiana State University, Baton Rouge, LA 70803

S. S. Iyengar
Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803

The "minimum vocabulary problem" for a dictionary has applications in indexing and other domains of information retrieval. A simple directed-graph model of a dictionary results in a linear-time algorithm for this problem. Since it is known that many minimum vocabularies can exist for a dictionary, a computationally useful criterion for finding a "desirable minimum vocabulary" is suggested and an $O(|V|^3)$ algorithm is outlined, where $|V|$ is the number of words in the dictionary, duplicates eliminated. Furthermore, some enrichments to the model and the computational intractability of a variant called the 1-lexicon problem are discussed. Directions for further work are indicated.

I. Introduction

Formal approaches have been extensively used for many useful problems in library and information science. In particular, graph-theoretic techniques are employed for content analysis of documents [1–3], citation analysis [4–6] and clustering of documents [7, 8]. In this paper we apply some well-known graph-theoretic techniques to solve the *minimum vocabulary* problem for a dictionary. A dictionary can be visualized as consisting of a set of words D, some defined in terms of the others and the rest not defined in terms of the others.

The minimum vocabulary problem is to obtain a set of words M such that (a) every word in D can be defined (or equivalently, understood) in terms of the words in M, and (b) M has the least cardinality among all sets having property (a).

As a terminological aside, it should be noted that the term "dictionary" does not necessarily mean a dictionary for a natural language, though it is helpful to view it in that sense.

Received March 6, 1986; revised May 21, 1986; accepted June 9, 1986

The term *minimum vocabulary* has been used by Moss [9], Sharp [10] and others, and their definition can be seen to be equivalent to ours. Further, Taulbee [2] has pointed out that no formal method to obtain a minimum vocabulary is known. It is interesting to note that Ogden (see [9]) has found a set of about 850 words which forms a minimum vocabulary of his basic English.

A minimum vocabulary can find applications in text processing and in the reduction of the size of an indexing vocabulary of documents for retrieval purposes. By text processing we mean content analysis of documents together with assignment of terms from the indexing vocabulary. For a particular subject, the index terms are collected from the relevant documents. Since every term in the dictionary can be expressed using the terms in the minimum vocabulary, we suggest that the minimum vocabulary be used as the indexing vocabulary. Now, for retrieving documents the set of search terms are first mapped onto the terms in the minimum vocabulary and the retrieval proceeds as usual. If the indexing vocabulary is not minimum, a greater retrieval effort may be needed. This is because the search would be carried out using a greater number of terms, in general.

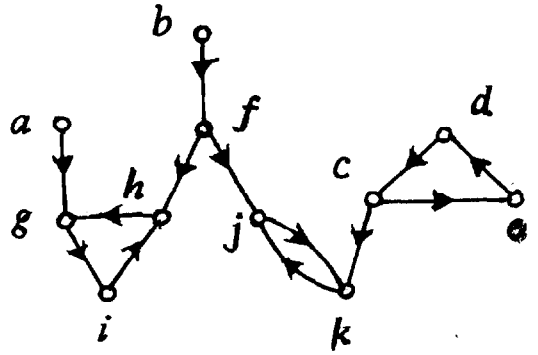
In this article we propose a simple directed-graph model for a dictionary and show that the minimum vocabulary can be obtained algorithmically by determining the *basis* of this graph. The algorithms used are well known in the graph-theoretic literature. The complexity of the algorithm is of the order of the "size" of the dictionary, which is optimal. Since it is known that many minimum vocabularies can exist for a dictionary, there is a need to see if any minimum vocabulary is better than the others in some sense. Towards this end, we give a natural and computationally useful criterion called the *access coefficient* of a minimum vocabulary, to help choose a *desirable minimum vocabulary*. Further, in a dictionary for a natural language, a minimum vocabulary corresponds to a set of words by which all meanings of all the words can be understood. Since the minimum vocab-

ulary may be a large set and is not always needed, it is useful to try to obtain a set of the minimum number of words which provide at least one meaning of every word whose meaning can be accessed in the dictionary. Unfortunately, we prove that this problem is NP complete, that is, no polynomial-time algorithm is likely to exist [11].

The paper is organized along the following lines. Section II contains the relevant graph terminology, the directed-graph model and a simple algorithm to find the minimum vocabulary. Section III describes some extensions to the model and the NP completeness of a variant called the *1-lexicon* problem. Conclusions are given in Section IV.

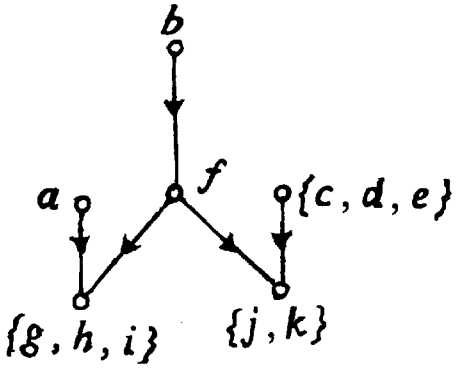
II. Directed-Graph Model of a Dictionary

We quickly review the pertinent definitions in graph theory needed for our purposes. A directed graph $G = (V, E)$ is a set of vertices V and a set of edges $E \subseteq V \times V$. The terms *directed path, path length, directed cycle or circuit, induced subgraph, indegree,* and *outdegree* have the standard definitions found in Harary [12]. For convenience, we use the term graph to mean directed graph and the vertex set and graph interchangeably. A vertex v is *reachable* from another vertex w, if there is a directed path from w to v. In Fig. 1 the vertex k is reachable from vertex b. A *strongly connected component* of a graph G is a subgraph H such that any two vertices of H are mutually reachable. The graph induced by $\{c, d, e\}$, $\{g, h, i\}$, $\{b\}$ are some of the strongly connected components of the graph G in Fig. 1. An *acyclic graph* is a graph having no directed cycles. The graph G^* shown in Fig. 2 is an acyclic directed graph. In general, a graph can be disconnected. Let $S_1, S_2, \ldots, S_k$ be the strongly connected components of a graph $G = (V, E)$. The *condensation* $G^* = (V^*, E^*)$ of G has $V^* = \{S_1, S_2, \ldots, S_k\}$ and $E^* = \{(S_i, S_j): i \neq j$ and there is at least one edge from a vertex in S_i to a vertex in $S_j\}$. Every $S_i, i = 1, 2, \ldots, k$ is called a supervertex. The graph G^* is the condensation of G and vertices $\{c, d, e\}$, $\{g, h, i\}$, $\{j, k\}$, $\{a\}$, $\{f\}$, $\{b\}$ are the supervertices of graph given in Fig. 2. A *basis* of a graph G is a set containing minimum number of vertices of G such that every vertex of G can reach a vertex of this set. Trivially, every vertex is reachable from itself. Note the change in the definition of the term "basis" as found in [12]. It can be seen that the following propositions hold:



G

FIG. 1.



G*

FIG. 2.

PROPOSITION 1: Every acyclic directed graph G contains at least one vertex of indegree 0 and a vertex of outdegree 0.

In Fig. 2 vertex $\{a\}$ has indegree 0 and vertex $\{j, k\}$ has outdegree 0.

PROPOSITION 2: The basis of an acyclic directed graph is the set of vertices with outdegree 0, and, moreover, the basis so obtained is unique.

The basis of the graph G^* in Fig. 2 is $\{\{g, h, i\}, \{j, k\}\}$.

PROPOSITION 3: The basis of a directed graph G is a set of vertices obtained by choosing exactly one vertex from each of the supervertices forming the basis of the condensed graph G^*. Furthermore, the cardinality of a basis so obtained is the minimum, in the sense that no other basis has a lesser cardinality.

For instance, a basis of the graph G is $\{g, h\}$.

The above results can be found, stated differently, in [12]. In the next section we formally define the dictionary and the directed-graph model.

A. Minimum Vocabulary of a Dictionary

We represent a dictionary D as follows:

$$D = \left\{ \begin{matrix} s_1 \rightarrow t_1^1, t_1^2, \ldots, t_1^{l_1} \\ s_2 \rightarrow t_2^2, t_2^2, \ldots, t_2^{l_2} \\ \cdot \\ \cdot \\ \cdot \\ s_j \rightarrow t_j^1, t_j^2, \ldots, t_j^{l_j} \end{matrix} \right\},$$

where $S = \{s_1, s_2, \ldots, s_j\}$ is a set of words called *source* and $\{t_p^q: p = 1, 2, \ldots, j, q = 1, 2, \ldots, l_p\} = T$ is a set of

words called *targets* such that $D = S \cup T$. Intuitively, for every source word s_p the definitions of s_p appear on the right-hand side, in terms of t_p^q, $q = 1, 2, \ldots, l_p$ for all $p = 1, 2, \ldots j$. The minimum vocabulary problem for D is to find a set of minimum number of words M such that every word of D can be defined in terms of some word in M. The *size* of the dictionary D is the cardinality of the set obtained by taking the disjoint union of S and T, where S and T are considered multisets. Note that multiset is a set in which duplicates of an element are allowed and disjoint union is the usual set union, but duplicates are not eliminated.

A directed graph G for a dictionary D is $G = (V, E) = T(D)$, where $V = D$ and $E = \{(s_p, t_p^q) : p = 1, 2, \ldots, j$ and $q = 1, 2, \ldots, l_p\}$. This means that for every source word u a directed edge goes to every target word w on the right-hand side of u. Then, accessing a definition of a word can be thought of as traversing along the directed edges, starting from the vertex corresponding to the word, until a known word is reached. Now we state the main result for finding the minimum vocabulary of D.

THEOREM 1: The minimum vocabulary of a dictionary D is a basis B of the graph $G = T(D)$, defined above.

PROOF: Let $u \in D$. Then there exists a word $v \in B$ such that v is reachable from u. This is because v is an element of the basis. In other words, u can be defined in terms of v. So, every word in the dictionary can be defined by words in B. Furthermore, B has the minimum cardinality by Proposition 3.

The following algorithm finds the minimum vocabulary of D.

Algorithm MINVOC (D: dictionary).

Step 1: Construct the directed graph $G = T(D) = (V, E)$ as defined earlier.
Step 2: Find the strongly connected components of G.
Step 3: Obtain the condensed graph $G^* = (V^*, E^*)$.
Step 4: Find the supervertices of outdegree zero and choose a word from each of the supervertices. Call this set B. B is a minimum vocabulary.

The correctness of the algorithm follows from Theorem 1 and Proposition 3.

We consider the following dictionary D as an example and illustrate the working of the algorithm MINVOC on D:

$$D = \begin{cases} a \rightarrow g \\ b \rightarrow f \\ c \rightarrow e, k \\ d \rightarrow c \\ e \rightarrow d \\ f \rightarrow h, j \\ g \rightarrow i \\ h \rightarrow g \\ i \rightarrow h \\ j \rightarrow k \\ k \rightarrow j \end{cases}$$

(1) The graph G shown in Fig. 1 represents the dictionary D.
(2) The strongly connected components are the graphs induced by $\{a\}$, $\{b\}$, $\{f\}$, $\{g, h, i\}$, $\{j, k\}$, and $\{c, d, e\}$.
(3) The condensation G^* of G is given in Fig. 2.
(4) The supervertices with outdegree 0 are $\{g, h, i\}$ and $\{j, k\}$. Let $B = \{g, k\}$ be a minimum vocabulary.

The adjacency list for G can be found out in time $O(|D|)$, where $|D|$ is the size of the dictionary as defined before. Therefore, Step 1 has $O(|D|)$ time complexity. Steps 2, 3, and 4 can be done in $O(|V| + |E|)$ time [13]. Since $O(|V| + |E|) = O(|D|)$, the time complexity of the algorithm is $O(|D|)$.

III. Desirable Minimum Vocabulary and NP Completeness of 1-Lexicon Problem

As we pointed out earlier, under the interpretation of dictionary as one for a natural language like English, our model is inadequate. In general, every target can be a group of words. We classify two types of such groups of words:

(1) *Denotational* group of words, where the meaning of the group can be understood by knowing each of its constituent words. For example, (*a feeble or intermittent light*) is a denotational group of words defining the word *glimmer*.
(2) *Nondenotational* group of words, where the meaning of the group can only be understood by considering it as a whole as a new word itself (such groups are commonly known as a phrase). (*to blow one's top*) is a nondenotational group of words.

Single-word targets can be treated as belonging to either category. But it might be realistic to treat such words as belonging to the denotational group. Under this generalized setting, we merely outline the procedure to get a minimum vocabulary. We construct the directed graph as follows. For every target which is a denotational group an edge goes from its source to every member of the group. Every nondenotational group of words is coalesced into a distinct *new word* and an edge goes from its source word to this new word. If a group of words contain both denotational and nondenotational components, a combination of the above two is carried out. It is easy to verify that applying algorithm MINVOC to the graph obtained as above, we get the required minimum vocabulary.

Since there can exist many minimum vocabularies, it is useful to find out if a particular minimum vocabulary is "better" than any other in some sense. We suggest a simple and natural criterion to select a *desirable minimum vocabulary*. This is particularly relevant if the dictionary represented in the form of a directed graph is to be used for retrieval purposes. The criterion used is based on the notion of minimizing the number of overall lookups for finding the meaning of word until a word in a minimum vocabulary is reached.

Let $S_1, S_2, \ldots, S_l$ be the supervertices of outdegree 0 in G^*. All $S_i, i = 1, 2, \ldots, l$ can be considered as sets, whose elements are words. A minimum vocabulary is an element of the set $S = S_1 \times S_2 \times \ldots \times S_l$. Again, we consider

every basis as a set, rather than as a tuple. Let the set of all minimum vocabularies be $\{B_1, \ldots, B_m\} = S$. Clearly, $m = |S_1| \times |S_2| \times \ldots \times |S_l|$. Let $x \in D - B_i$. Then we define $R(x, B_i)$ as the set of words in B_i reachable from x. Let $b_i \in R(x, B_i)$ and $d(x, b_i)$ represent the minimum path length from x to b_i. The *access coefficient* of a minimum vocabulary B_i is defined as $AC(B_i) = \sum_{x \in D - B_i} \sum_{b_i \in R(x, B_i)} d(x, b_i)$. The desirable minimum vocabulary is a basis B_j such that $AC(B_j) \leq AC(B_i)$, $i = 1, 2, \ldots, m$. Let $S_i = \{s_i^1, \ldots, s_i^{|i|}\}$, for all $i = 1, 2, \ldots, m$. The aim is to find out a set $V = \{s_1, \ldots, s_m\}$ such that $s_i \in S_i$, $i = 1, \ldots, m$ and $AC(V) \leq AC(M)$ for all minimum vocabularies M. Now, we outline the steps needed to obtain a desirable minimum vocabulary. Let $U = \cup_{i=1}^{m} S_i$.

(1) For every word b in U, find the sum of distances $D(b)$ from all words reaching b.
(2) In any supervertex S_i, take the word having the minimum $D(b)$ over all b in S_i, as an element of the desirable minimum vocabulary V.
(3) Do (2) given above, for all the supervertices. The words so chosen comprise V.

In fact, one can find the shortest distance between each pair of vertices and select the relevant distances needed for the steps (1) and (2) above. An algorithm for finding the desirable minimum vocabulary using the above ideas has $O(|W|^3)$ time complexity, where W is the set of words in D, duplicates eliminated. Note that the access coefficient as we have defined is a simple criterion and open to further improvements or changes.

Now we work out the steps for obtaining the desirable minimum vocabulary for the dictionary D given in Section II, part A.

1. The vertex $\{g, h, i\}$ is a supervertex. Vertex g is reachable from a, b, h, and i. The distance of g from a, b, h, and i is 1, 3, 1, 2, and 2, respectively. The sum of these distances is equal to 9. Similarly, h, i, j, and k have their sum of distances from other vertices equal to 9, 12, 13, and 11, respectively.
2. Vertices g and h in the supervertex $\{g, h, i\}$, have their sum of distances equal to 9 which is a minimum. Therefore either g or h is chosen to be an element of the desirable minimum vocabulary. Similarly, vertex k with its sum of distances equal to 11 is chosen from the supervertex $\{j, k\}$ as an element of the desirable minimum vocabulary.
3. The desirable minimum vocabulary is either $\{h, k\}$ or $\{g, k\}$.

We know that the minimum vocabulary B for a dictionary helps understand every definition of every source word. Since B could be a huge set in general and may not be needed for all purposes, we study the following problem:

1-Lexicon Problem.

INSTANCE: Given a dictionary $D = S \cup T$, represented as in Section II, and a positive integer k.

QUESTION: Does there exist a set of words $B \subseteq D$ such that $|B| \leq k$ and every source word can have *at least* one word in B serving as its meaning?

In Fig. 3, we have a graph representation for a dictionary D. The minimum vocabulary for D is $\{c, d, e\}$. For the 1-lexicon problem it is enough to know the word d to know one definition of every source word.

Note that the 1-lexicon problem is posed as a decision problem, that is, as a problem having a "yes" or "no" solution. In general, we may be more interested in finding a solution as a collection of objects, rather than a mere "yes" or "no." For example, in the 1-lexicon problem we could be interested in finding the set B with $|B| \leq k$, if it exists. Such problems are known as the *optimization problems* as opposed to their decision versions. The main idea behind studying decision problems is that if the decision problem itself is *computationally hard*, then the corresponding optimization problem is at least as hard. So, by showing that the 1-lexicon problem is NP complete we are actually showing that finding the set B whose cardinality is at most k is as hard. All NP-complete problems are known to have only algorithms with exponential time complexity, that is, computationally expensive algorithms. Informally, a problem X is *complete* for a class of problems C, if $X \in C$ and X is computationally as hard as any other problem in C. In other words, X is one of the hardest problems in the class C. Hence the NP-complete problems are the hardest problems in the class NP. We can say that the class NP contains most of the important problems which are useful from a practical point of view. Though it would be beneficial for the reader to have a background in the theory of NP completeness (Garey and Johnson [11]), we have tried to make the presentation self-sufficient.

Formally, NP is the class of decision problems that can be solved by polynomial-time algorithms on a nondeterministic Turing machine. The implication of a problem X being NP complete is that no efficient algorithm, i.e., polynomial-time algorithm, is likely to exist for X. In order to show that a problem X is NP complete, (i) X is shown to be in NP, and (ii) A polynomial-time transformation f is shown to take a generic instance I of a known NP-complete problem Y to an instance $f(I)$ of X such that I has a "yes" solution if and only if $f(I)$ has a "yes" solution. Now, we show that the 1-lexicon problem is NP complete, by showing that a restricted version of the 1-lexicon problem called the *disjoint 1-lexicon* problem is NP complete. The disjoint

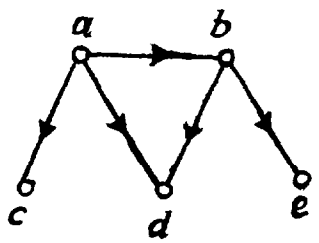


FIG. 3.

1-lexicon problem is the same as the 1-lexicon problem with the additional restriction that $S \cap T = \phi$. Note that the disjoint 1-lexicon problem has been introduced here more as a tool to prove the NP completeness of the 1-lexicon problem than as an important problem in itself. Now, we state the result and indicate the proof.

THEOREM 2: The disjoint 1-lexicon problem is NP complete.

PROOF: Trivially, the disjoint 1-lexicon problem is in NP. The polynomial-time transformation is from the ***hitting-set*** problem which is NP complete (see [11]).

In the ***hitting-set*** problem, A is a set and C is a collection of subsets of A. Given a positive integer k, the problem is to find out whether there exists a subset B of A such that every set of C has at least one element in B and that the cardinality of B is at most k.

Let $A = \{a_1, \ldots, a_p\}$ and $C = \{C_1, \ldots, C_q\}$ be an instance of the ***hitting-set problem***. Here, all C_i's are subsets of A. Construct an instance of the disjoint 1-lexicon problem as follows: In the disjoint 1-lexicon problem, the set $S = C$ and $T = A$ and for every source "word" $C_i \in S$, $i = 1, 2, \ldots, q$, the "targets" are $a_i^1, a_i^2, \ldots, a_i^{m_i}$, where the set $\{a_i^1, a_i^2, \ldots, a_i^{m_i}\}$ equals C_i. The positive integer k in both the problems is the same. The reader may verify that this transformation is indeed polynomial-time computable and a "yes" solution for the ***hitting-set*** exists if and only if the instance got ten from the transformation has a "yes" solution.

We illustrate the polynomial transformation from the ***hitting-set*** problem by means of an example. Let $A = \{c, d, e\}$ and $C = \{\{c, d\}, \{d, e\}\}$ constitute an instance of the ***hitting-set*** problem. The following instance of a dictionary D is produced after applying the polynomial transformation given above:

$$D = \begin{Bmatrix} \{c, d\} \longrightarrow c, d \\ \{d, e\} \longrightarrow d, e \end{Bmatrix}.$$

IV. Conclusions

We proposed a simple directed-graph model by which the minimum vocabulary problem was shown to have a linear-time algorithm. Furthermore, some extensions and the NP completeness of a variant of the minimum vocabulary problem were discussed. The apparent limitations of our model are with regard to its use for a natural-language dictionary. In general, a language like English, will have the same word appearing in different tenses. According to our model, each such word would be identified as a distinct word. This would result in an increase of memory and a possible increase in the size of the minimum vocabulary. But this can be taken care of by some suitable "higher-level" actions. The main use of the model lies in the reduction of the size of the index vocabulary for documents pertaining to a field and syntactic text manipulations. Identifying more useful criteria for finding out a desirable minimum vocabulary needs further investigation.

Acknowledgments

The authors would like to thank Dr. Bert Boyce, Dr. Danny Wallace, Dr. Donald Kraft, and Mr. Nageswara Rao for their valuable suggestions and comments. We thank the referees for their useful comments which helped improve the contents of the article.

References

1. Gross, P. "A Method of Using Theory of Graphs for the Content Analysis of Automated Document Reference Store." *I. Informatik*, **228**(6): 12–14; 1981.
2. Taulbee, O. E. "Content Analysis, Specification, and Control", *Annual Reviews of Science and Information Technology*. **105**; 1968.
3. Kropp, D.; Walch, G. "A Graph Structured Text Field Index Based on Word Fragments." *Information Processing and Management*. **17**(6): 363–376, 1981.
4. Shaw, W. M., Jr. "Critical Thresholds in Co-Citation Graphs." *Journal of the American Society for Information Science*. **36**(1): 38–43; 1985.
5. Garner, R. "A Computer-Oriented Graph Theoretic Analysis of Citation Index Structures." *Drexel Information Science Research Studies*. Drexel Press, 1967: 3–46.
6. Garner, R. "Graph Theory as an Information Retrieval Tool—An Example of Citation Indexing." *American Documentation Institute Proceedings, Fourth Annual Meeting*; 1967: 80–83.
7. Ogilvie, J. C.; Olson, C. L. "On the Use of Complete Subgraphs in Cluster Analysis." *Information Processing Letters*. **1**(3): 76–79; 1972.
8. Willett, P. "Document Retrieval Experiments Using Indexing Vocabularies of Varying Size. II Hashing, Truncation, Digram and Trigram Encoding of Index Terms." *Journal of Documentation*. **35**(4): 296–305; 1979.
9. Moss, R. "Minimum Vocabularies in Information Indexing." *Journal of Documentation*. **23**(3); 1967.
10. Sharp, Jr., "Where to go from Here?" *ASLIB Proceedings*. **23**(1): 33–46; 1971.
11. Garey, R.; Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freeman; 1979.
12. Harary, F. *Graph Theory*. Reading, MA: Addison-Wesley; 1972.
13. Tarjan, R. "Depth-first Search and Linear Graph Algorithms." *SIAM Journal on Computing*. **1**(2): 146–160; 1972.