VOCNETS - A TOOL FOR HANDLING FINITE VOCABULARIES

Hans KARLGREN
KVAL Institute for Information Science
Södermalms torg 8
S-116 45 Stockholm
Sweden

Jürgen KUNZE
Academy of Sciences of the GDR
Prenzlauer Promenade 149-152
Berlin, DDR - 1100
German Democratic Republic

Abstract

A method is proposed for storing a finite vocabulary in a manner which makes it convenient to recognize words and substrings of words. The representation, which can be generated automatically from a list of words or from given representations of other sets by means of which the vocabulary has been defined through set or string operations, has the form of a modified finite-state grammar, a form eliminating the multiplicative effects of conjunction, complementation, etc., on the node sets of conventional finite-state representations.

0. Background

Traditionally, linguists describe sentences, and inflected and derived word forms by means of rules, whereas vocabularies are accounted for by enumeration. But even for the purpose of specifying a given lexicon or the vocabulary of a given piece of text we find mere enumerations inconvenient to access and not very illuminating. We want answers to be readily given to questions like whether a given string is a member (or a prefix, a suffix, some other substring or sequence of substrings of a member), or which elements of some set of strings have such properties. That is, we want to arrange the lexical data so that it is easy to perform Boolean and string operations on sets of words.

We therefore introduce a grammar-like representation for a finite vocabulary, specifying it as is, i.e., without exaggeration or omission, with no claim on the linguistic status of the set described or the rules constructed to specify it. No prediction about potential strings outside the given set is suggested. The representation can be algorithmically derived from a list of the words in the vocabulary.

The proposed tool appears to have theoretical as well as computational merits.

1. Task

We thus require a method for representing a v o c a b u l a r y V of strings over an a l p h a b e t A (of letters, phonemes, morphemes or other atoms), where
* A is small compared to the vocabulary V (say, 30 against 30 000 or 300 000),
* the vocabulary V, though large, is finite,
* V has a "structure" in the sense that, typically, a string in V contains substrings included in other strings in V.

We want the representation to
* permit convenient r e t r i e v a l of strings and substrings of strings in V,
* be algorithmically constructed on s u c c e s s i v e i n p u t of strings in V, or, if V is defined through B o o l e a n or s t r i n g o p e r a t i o n s on other sets, be derivable from operations on representations of these more elementary sets,
* be reasonably c o m p a c t for practical computational applications.

2. Modified Finite-State Representation

We have chosen to represent vocabularies as modified finite-state grammars, which we shall call vocnets.

A vocnet will include a finite directed graph with edges, a r r o w s, labelled with elements of the alphabet A. Such a graph will specify a vocabulary over the alphabet A if we mark a subset S of the nodes as source nodes and define as an accepted word the concatenation of the labels of such paths through the graph from nodes in S as arrive under certain side conditions at a set of nodes which fulfills given target conditions.

We do not assume a vocnet to be deterministic in the sense that for any node i and string α there exist only one node j such that α is a path from i to j. Should we introduce such a restraint, it can be proven that it is lost already under regular operations on the vocabularies, i.e., that this attractive feature will be absent from a vocnet derived in the manner we propose for the union, concatenation set or closure of the vocabularies, for which deterministic vocnets had been introduced.

Precautions had to be taken to keep the mechanically generated representations compact. In particular, it was essential to eliminate the well-known multiplicative effect on the number of states arising when standard finite-state grammars are combined by intersection and complementation.

3. Definition of vocnet graphs

A vocnet graph $U = \langle A, N, C', C'' \rangle$ is a quadruple, where

A is an alphabet of a t o m s a, b, c, ...

N is a set of n o d e s h, i, j, k, ...

C' and C" are mappings of A into $N \times N$.

We define $C(x) = C'(x) \cup C''(x)$ as the set of c a t e g o r i e s of the atom x.

We define the product $C_1 \circ C_2$ of two category sets C_1 and C_2 as

$$C_1 \circ C_2 = \{(i, j) \mid \exists k\ (i, k) \in C_1 \wedge (k, j) \in C_2\}$$

and the category set for a string $\alpha = x\beta$ as

$$C(\alpha) = C(x) \circ C(\beta)$$

We shall say that the atom x c o n n e c t s the set M1 to the set M2 in U iff

either M2 is the set of all j for which there is a node i in M1 such that $(i, j) \in C'(x)$,

or M2 is the set of all j for which there is a node i in M1 such that $(i, j) \in C''(x)$.

We shall also say that a string $\alpha = x\beta$ connects M1 to M2 if there is some set M3 such that x connects M1 to M3 and β connects M3 to M2.

By introducing two kinds of arrows, one can so to speak synchronize parallel paths: the restraint that in every path the arrow associated with one position in a string will have to be of the same kind can be utilized to partition the graph into zones which correspond to segments of the strings, if one kind of arrows, i n t r a z o n e arrows (those in C') join nodes within the same zone and another kind, i n t e r z o n e arrows (those in C"), join nodes in one zone with nodes in another zone. A string can then be seen as consisting of segments separated by junctures, where each segment is associated with parallel intrazone arrow sequences and each juncture with parallel interzone arrows.

4. Definition of Vocnets

A vocnet G is a triple <U, S, P>, where $S \subseteq N$ is a non-empty set of s o u r c e nodes
P(M) is a t a r g e t c o n d i t i o n on node sets M, P(M) being a proposition over elementary conditions of the form that M overlaps with some subset E of N, say

$$(M \cap E1 \neq \emptyset) \wedge \sim(M \cap E2 \neq \emptyset).$$

The sets E1 and E2 here form the t a r g e t a r e a s of G.

The union of all minimal sets M for which P(M) is true in the vocnet G will be called the t a r g e t s e t T of G.

A vocnet G defines the language L(G):

$\{\alpha \mid \exists M \subseteq N$ and α connects S to the non-empty node set M and P(M) is true$\}$

Whereas for a string to be accepted by a conventional finite-state grammar it is enough that it is associated with one permitted path through the graph, a string will be accepted by a vocnet if it is associated with a set of simultaneous paths, each leading from a source node to a target node, these target nodes forming a permitted combination M (i.e., M is not empty and P(M) is true).

The vocnet may contain special e x i t c h e c k e r s. An exit checker is a dummy zone, consisting of exactly one node connected to itself by an arrow in C' for each atom in A. By using exit checkers, local conditions for zones can be accounted for in the target conditions for the whole vocnet. The exit checkers, in a way, will then freeze the zone exit conditions so that they remain accessible for verification when the whole graph has been passed through.

5. Generation of Vocnets from List of Words

A vocnet for a given vocabulary can be generated algorithmically in the following manner.

Words are entered one by one. For each new word unique new nodes are introduced: if the new word is $x_1 x_2 \ldots x_n$, each letter x_m is given the new category (k_r, k_{r+1}), where no k_φ existed before.

Clearly, this procedure will create a vocnet which will account for all and only the words given. The set of nodes, however, will typically be much larger than necessary, but it can be reduced - after one word has been entered or after the insertion of several words - by appropriate fusion of nodes; cf. section 8 infra.

6. Set Operations on Vocabularies

In the following, it will be assumed that the vocabularies considered are strings over the same alphabet A, that none of them includes the empty string, and that the vocnet graphs which we combine have disjunct sets of nodes.

6.1 Complement Formation

Given a vocnet G1 for a language L1, the vocnet G for the complement L is given immediately by replacing P1 by its negation

$G = \langle U1, S1, \sim P1 \rangle$,

if G1 is complete in the sense that for any string there exists some path beginning in an element of S1. If G1 is not complete in this sense, it can be made complete at the expense of adding one more node.

6.2 Union

In a vocnet $G = \langle U, S, P \rangle$ for the union of L(G1) and L(G2) the vocnet graph U is formed directly through union of the elements of U1 and U2, and P is formed through disjunction:

$U = \langle A, N1 \cup N2, C1' \cup C2', C1" \cup C2" \rangle$
$S = S1 \cup S2$
$P(M) \Leftrightarrow P1(M) \vee P2(M)$ for $M \subseteq N$.

6.3 Intersection

In a vocnet G for $L(G1) \cap L(G2)$, U and S are formed as in the case of union and

$P(M) \Leftrightarrow P1(M) \wedge P2(M)$ for $M \subseteq N$.

Thus, one and the same vocnet graph will serve as a component in vocnets defining different languages.

7. String Operations on Vocabularies

7.1 Concatenation

The concatenation set V of V1 and V2, i.e., the set V of strings consisting of a string in V1, specified by the vocnet G1, concatenated with one in V2, specified by the vocnet G2, is defined by a vocnet G

$G = \langle U, S1, P \rangle$
where
$U = \langle A, N1+ \cup N2, C1' \cup C2', C1"+ \cup C2" \cup C12" \rangle$
$P(M) \Leftrightarrow Q1(M) \wedge P2(M)$

Here
N1+ is N1 with the addition of exit checkers: if G1 has the target areas E1, E2, ..., N1+ will contain the exit checkers f1, f2, ...,

C1"+ is C1" with the addition of arrows for each atom from each node in Ep to the exit checker fp,

C12"(x) is the set of all arrows (i, j) with $i \in T1$ and $j \in N2$ for which $(h, j) \in C1'(x)$ for some $h \in S2$.

Q1(M) is the frozen version of P1(M), with f1, f2, ..., substituting E1, E2, ...

The vocnet graphs U1 and U2 have thus been integrated as zones into the new vocnet graph. A few exit checkers have been added

to permit expressing the restraints on the passage through the zone U1 as target conditions on the totality of G. Thanks to the use of exit checkers the complexity of the target condition P of G in terms of the number of target areas is not the product of the complexities of P1 and P2 but less than their sum.

7.2. Restricted Iteration and Involution

The languages $L(G1) \cup L(G1)^2 \cup \ldots \cup L(G1)^q$

and $L(G1)^q$ $(q \geqq 2)$ may be represented as vocnets that are constructed in a similar way as for concatenation, with G1 in the role of G2, but the exit checkers have to be stratified so that we may count the depth d of the concatenation. Therefore C"(x) contains besides the categories explained in 7.1 all pairs $({}^{d}fp, {}^{d+1}fp)$ for $1 \leqq d \leqq q-1$.

The target condition for restricted iteration is

$$P(M) \Leftrightarrow P1(M) \wedge (M \cap \{{}^{q}f1, {}^{q}f2, \ldots\} = \emptyset) \wedge$$

$$({}^{q-1}P1(M) \Rightarrow \ldots \Rightarrow {}^{1}P1(M))$$

and for the p-th power of L(G1)

$$P(M) \Leftrightarrow P1(M) \wedge (M \cap \{{}^{q}f1, {}^{q}f2, \ldots\} = \emptyset) \wedge$$

$${}^{q-1}P1(M) \wedge \ldots \wedge {}^{1}P1(M).$$

Here, ${}^{d}P1(M)$ are the frozen stratified target conditions of G1.

7.3. Decatenation

Given one vocnet G1 (say for words beginning with a prefix) and another vocnet G2 (say for prefixes and prefix sequences), we search a vocnet G (say for words stripped of their prefixes) such that $\alpha \in L(G)$ iff

$$\exists \alpha_1 \exists \alpha_2 (\alpha_1 \in L(G1) \wedge \alpha_2 \in L(G2) \wedge$$

$$\alpha_2 = \alpha_1 \alpha)$$

The following vocnet G will satsify our requirement:

G = <U1, S, P1>

where S is the union of all sets $M \subseteq N1$ for which S1 is connected to M in G1 by some string contained in L(G2).

8. Equatability and Node Fusion

Vocnets generated with the incremental algorithm described in section 5 above typically contain more nodes than a minimal vocnet for the same language. Similarly, vocnets derived from other vocnets tend to be highly redundant.

Compacting of a given vocnet can be algorithmically performed as follows.

We shall say that nodes in a vocnet G are **equatable** if they can be identified without affecting the language defined by G.

The following definitions permit us to find pairs of equatable nodes.

We first define some equivalence relations between nodes.

The nodes i and j are **precedence equivalent** in a vocnet graph U iff for all k and x

$$(k, i) \in C'(x) \Leftrightarrow (k, j) \in C'(x)$$

and

$$(k, i) \in C"(x) \Leftrightarrow (k, j) \in C"(x)$$

The nodes i and j are **succession equivalent** in a vocnet graph U iff for all k and x

$$(i, k) \in C'(x) \Leftrightarrow (j, k) \in C'(x)$$

and

$$(i, k) \in C"(x) \Leftrightarrow (j, k) \in C"(x)$$

The nodes i and j are **source equivalent** in a vocnet G iff

$$i \in S \Leftrightarrow j \in S$$

The nodes i and j are **target equivalent** in a vocnet G iff for any subset M of N

$$P(M \cup \{i\}) \Leftrightarrow P(M \cup \{j\}).$$

Now the nodes i and j are **left equivalent** in a vocnet G iff they are precedence and source equivalent. They are **right equivalent** in a vocnet G iff they are succession and target equivalent. They are **equatable** if - but not necessarily only if - they are left or right equivalent.

By successive fusion of pairwise equatable nodes vocnets can be - not rarely drastically - compacted. It should be noted, however, that equatability is not an equivalence relation and that reduction of a given vocnet graph does not yield a unique result but depends on the choice of node pairs to identify in each step of the procedure.

9. Parasites

By **parasites** of a language L we shall mean strings which are not members of L nor substrings of members of L.

Clearly, if with the vocnet G the set $C(\alpha)$ is empty, α is a parasite of L(G): α is not a member nor will it become a member whatever is appended at either end.

We shall say a node i in a vocnet G is **genuine** if there is some string α associated with a path from a source node in G via i to a node in some M, such that α connects S to M and P(M) is true.

If all nodes in a notvec are genuine, a string α is a parasite iff $C(\alpha)$ is empty. The vocnet will then offer us an associative calculus for recognizing parasites (and strings which constitute the beginning of a word or the end of a word).

A node i is ingenuine if no path leads from nodes in S to i or from i to nodes of T. If P(M) has the simple form that M must overlap with some given target set, a node i is ingenuine only if the preceding condition is fullfilled.

10. Node Elimination

Ingenuine nodes can be removed from the graph U without affecting the language accepted by G = <U, S, P>.

Successive elimination of ingenuine nodes and fusion of equatable node may lead to considerable compression and simplification of a given vocnet. It should be observed that the final, irreducible result of such compression is not independent of the choice at each stage of what reduction operation to perform.