DISEMINER: A Distributional-Semantics Inference Maker[†]

Sheldon Klein[††], Stephen L. Lieman and Gary E. Lindstrom

Carnegie Institute of Technology
Pittsburgh, Pennsylvania

June 13, 1966

[†] This research was supported in part by Public Health Service Grant MH 07722 from the National Institute of Mental Health, and in part by the Advanced Research Projects Agency under the Department of Defense under Contract SD-146 to Carnegie Institute of Technology.

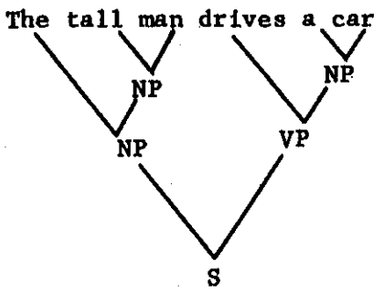
[††] Now holds a joint appointment in the Linguistics and Computer Sciences Departments at the University of Wisconsin, Madison, Wisconsin.

1.0 Introduction

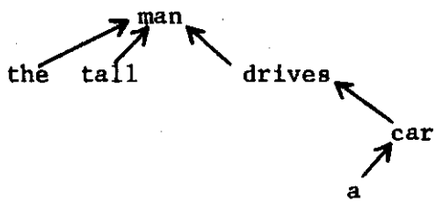
The purpose of the DISEMINER system is to explore the relation between lexical distribution criteria and semantics. It is hoped that the system, in its learning mode, will be useful in collecting data for deriving semotactic rules in a stratificational grammar [1]. The system, written in ALGOL 20 and operational on the G-21 computer, is capable of learning distribution classes of lexical items through the processing of text, and using distributional criteria to answer questions that are broader than the context of the text processed. The methodology follows a line of research that was considered, but never followed, in early work on the SYNTHEX project [2]. Distributional information is stored in terms of a dependency structure that differs from the SYNTHEX version in that dependency relations among stem types, rather than stem tokens, are stored in matrix format. That is, each stem is listed only once, and its dependency relations in all text processed by the system are associated with a single entry. (In the SYNTHEX system, separate dependencies are tabulated for each occurrence of a stem.) The stored relations include all possible transitive paths as well as direct ones. Because dependency analysis is weakly equivalent to phrase structure analysis [3], it is possible to view this data structure as a tabulation of the distributional potential of stems with respect to phrase structure criteria rather than criteria of linear contiguity.

2.0 Dependency and stratificational grammar

Whereas a phrase structure analysis depicts relations among units of varying size, in the form of a tree structure, e.g.



a dependency analysis indicates relations among units of the same size, in the form of a directed graph, e.g.



An excellent account of dependency theory can be found in the work of Hays [4].

If one assumes that under certain conditions dependency is transitive, it is possible to gain a measure of control over meaning. Several systems in the areas of information retrieval and automatic essay writing have been built incorporating such a principle [2, 5]. Basically, it is assumed that dependency is transitive across forms of the verb 'to be' and like tokens of the same noun. Thus it is possible to build a network connecting all the sentences in a text. The basic rule of semantics is that if the vocabulary and dependency relations in one text are included in the dependency relations of another, then the meaning of one is included in the other.

Rulon Wells and Zellig Harris provided early discussions of the relation of distributional criteria to the phrase structure grammar [6, 7, 8]. Harris discussed the relation of distributional criteria to meaning. In terms of a stratificational model of language [1], the meaning structure of a language is contained on a semantic stratum existing above a lexemic one. What meaning relations can be inferred from lexical distribution criteria are a reflection of the highly complex mapping of sememic structure onto lexemic. In the cases where such mappings are one to one, lexotactic criteria may be used as a substitute for semotactic relations.

In terms of grammatical analysis, the sememic relations can be determined by analysis of the lexemic; one assumes a one to one mapping between sememic and lexemic strata, except where pertinent data requires the positing of more complex mapping rules.

The DISEMINER system can be used as a tool for just such analysis. It can be viewed as a question answering machine that answers questions of the type, "Is it possible that ...?", where the answer is one of semantic plausibility as determined by inferences made on the assumption that the distribution classes are also semantic classes. Each case of a positive answer violating an informant's view of the possible is data for analysis of sememic entities and relations that do not conform to a one to one mapping onto the lexemic stratum. The system also can be used to answer Boolean questions and to obtain listings of rather refined semantic classes. (At the moment the working program limits such requests to queries of the type: $\left\{\begin{matrix}\text{Who}\\\text{What}\end{matrix}\right\}$ V1 (and) V2 (and) ... V_n.)

3.0 DISEMINER system

Inputs to the system may be of three types: text sentences, questions, and riddles. The complexity of the sentences must conform to limitations of the phrase structure dependency grammar.

A special feature of the system is that the dictionary is compiled from information supplied with the input text. An as yet unwritten routine would permit the optional omission of such data.

3.1 Data Organization

DISEMINER, viewed as a primitive information retrieval system, has the following tasks: to maintain a stem-type dictionary with provision for expansion both of stems and of parts of speech for existing stems, and to compile an efficient representation of the dependency network as deduced from the input text by the parser. With respect to current information retrieval systems, the objectives were modest, yet some effort was made to optimize the system with consideration to the criteria of storage allocation efficiency, data expandability, and network search time.

The stem-type dictionary is responsible for summarizing and directing the retrieval of stem types and their associated part of speech tokens. A stem type entry in this table is a collection of triples

$$< \pi, a, t >$$

where π is a part of speech number equal to the serial entry number in

the alphameric part of speech table; a is the alphameric string representing a word associated with that, and t is a boolean variable set TRUE if and only if that stem-part of speech token is designated transitive in the dependency network. The convention is made the the stem itself has part of speech zero, so the general format is that of a collection of such triples with the stem at the head, followed by subcollections of triples for each part of speech number actually occurring for that stem. In the interest of storage allocation efficiency, the table is actually chain linked, so that the triples for any particular stem are not necessarily contiguous but no gaps exist in the table for non-existing stem-part of speech types. This facilitates updating of the dictionary as well, so that DISEMINER can readily pass from input to question mode and back to input mode.

Since the stem dictionary is not linearly packed in memory, the accessing of data from it requires either a costly table scan or an access table with fixed entry size permitting indexing into the appropriate entry. The latter choice was made for DISEMINER, and the serial entry in this stem access table is the numerical representation for the stem throughout the system.

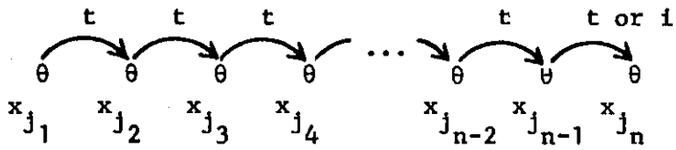
The parsing routine, to be described subsequently, provides as output from input text a collection of stem links

$$< x_j \quad \overset{t,i}{\rightarrow} \quad y_j > \; j = 1,\ldots,k$$

where the link is tagged t or i if the stem-part of speech token for stem x_j in the sentence under scan is denoted transitive or intransitive, respectively. If the parse is successful for a given input sentence, the

dependencies are stored in the set of three system dependency matrices T, I, and M in the following manner. For all links colored transitive, t_{x_j, y_j} is set to 1, for all links colored intransitive, i_{x_j, y_j} is set to 1, and for all links no matter what the tag m_{x_j, y_j} is set to 1. At the end of the input phase, therefore, the binary matrices T, I, and M contain all the dependency paths of length one that are determined to be respectively transitive, intransitive, and either transitive or intransitive.

In other dependency-oriented inference systems (such as SYNTHEX [2] of SDC), the data structure has been left at this stage (with some additional linkage back to the original text for sentence extraction), i.e., a set of binary links between stems. While this makes the sentence inputting fast, it demands extensive and redundant network searching for each question to be answered. DISEMINER's matrix representation facilitates the computation of all ultimate links, i.e., all links of the form



(1)

where $n \geq 2$.

It is a well-known property of graph matrices that if X is a binary matrix representing the immediate connectivity of nodes, then X^k represents the connectivity of nodes by paths of exactly length k. Since all paths must be of length less than or equal to the maximum number of nodes, the set of all paths is precisely the matrix $X^* = \bigvee_{k=1}^{d} X^k$, where the $\vee$ operator, as usual, denotes element-wise disjunction, and d is the dimension of

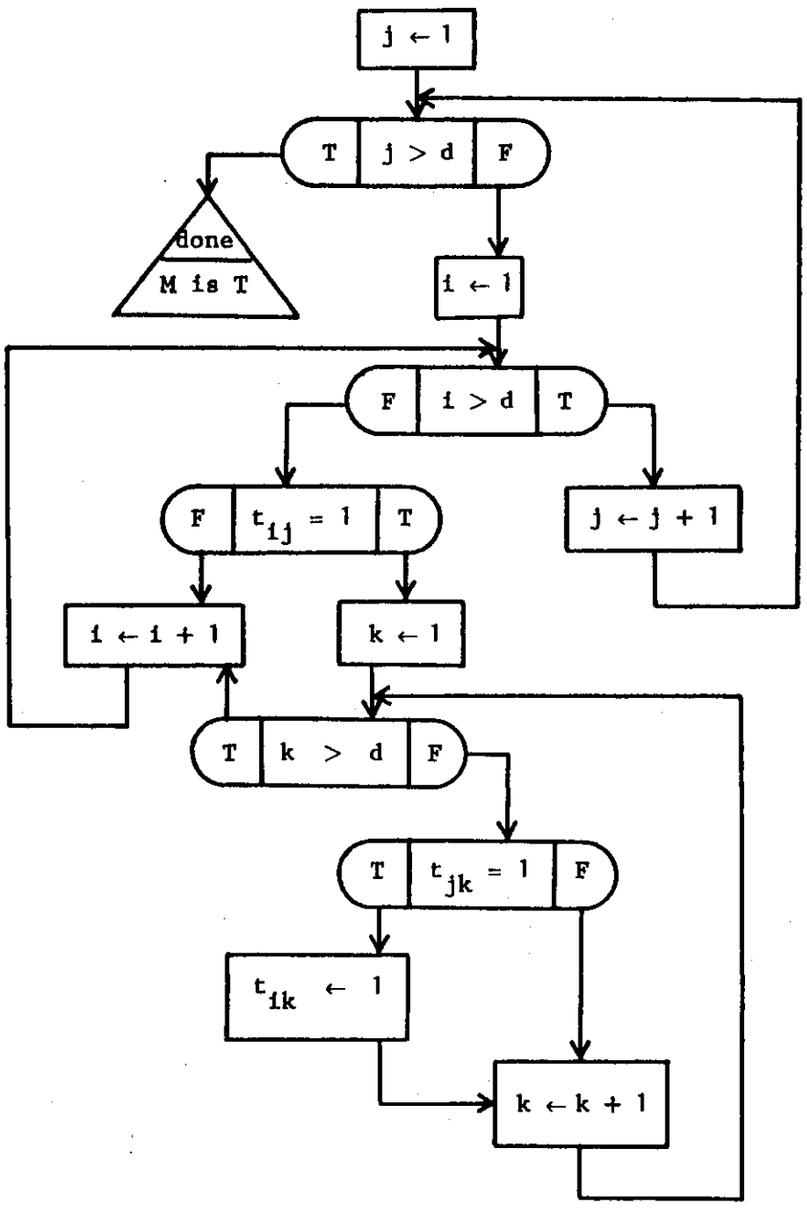
the square matrix X.

The set of all paths of the form shown in figure (1) may now be easily characterized. The transitive closure matrix, T^*, represents all paths of minimal length between two nodes using only transitive basic links. The final (optional) intransitive basic link is added by multiplying T^* by I, in intransitive basic link matrix. The final representation for the merged matrix M, therefore, is the sum of all transitive paths plus all transitive paths with a final intransitive link adjoined. That is,

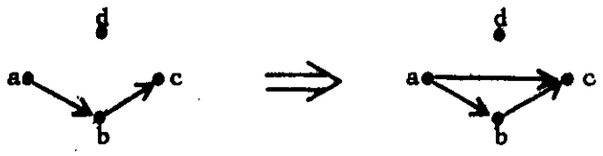
$$M = T^* I + T^* + I \qquad (2)$$

The straightforward method of calculating M by d matrix multiplications is a slow one, and furthermore increases in time as d^3. An alternative method published in 1962 by Stephen Warshall [9] behaves more on the order of d^2. This algorithm is particularly efficient for the dependency matrices in that for a given d its speed increases with the sparseness of the matrix, and typically a dependency matrix for short text is of low density.

The procedure is shown in the following flow chart:

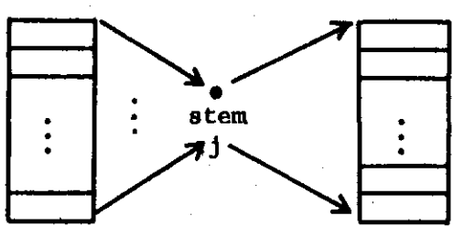


Briefly, the process is for each stem to form all paths of length two passing through the stem, counting paths as formed previously as paths of length one. For example,



Warshall provides a combinatorial proof that this procedure done once for all the stems in any order, forms all possible paths through the network. The reason the algorithm approaches behavior as d^2 for sparse matrices is that the innermost loop of the process is entered only if half the desired path is already know to exist, e.g. in the example above it is known that nothing can link from a through d since no link exists from a to d.

One further improvement can be made to decrease run time. Since the outermost iteration is through the stems letting all dual paths be formed, the process for each stem can be done by two table generations and the connections of all cross combinations.



This represents a time expenditure on the order of $d[2d+(md)^2] = 2d^2 + m^2d^3$, where m is the density of the final matrix. The non-tabular approach described previously and actually used in DISEMINER behaves as $d[md \cdot d] = md^3$. Since m is in the neighborhood of .1 for M, we have a theoretical approximate

breakeven point for the two methods of 22 stems, meaning the latter approach is preferable for dictionaries larger than 22 stems provided the table space is available.

3.2 Brief description of DISEMINER's routines

This section presents the major features of the system's routines.

1. Initialize

The dependency grammar and the part of speech table are read in. Array bounds are defined and numerical values are assigned to the parts of speech.

2. Input text sentence

The input sentences assume the following format:

$$S ::= A_{(1)}\ A_{(2)} \cdots A_{(N)}$$

where $A_{(i)} ::= word_{(i)}/stem_{(i)}/part\ of\ speech_{(i)}$. $i = 1, 2, \ldots N$.

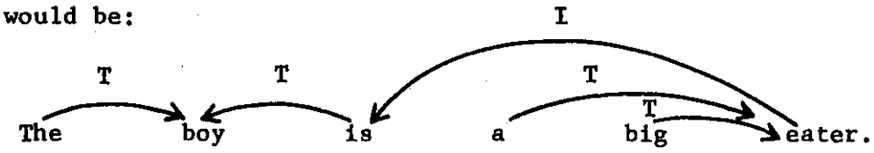
E.g. the/the/Art boy/boy/N eats/eat/V fish/fish/N .

The alphameric representations of each word are stored in a word list table. The stems are converted to stem numbers via table look-up in the list of current stems. If a stem has already occurred during the run, it is assigned its previous value. If a stem is not in the system then it is added to the system and given the next highest integer value stem number. The stem numbers are stored in a stem list table for future reference. The parts of speech are converted to part of speech numbers and stored in the integer array Sentence. Illegal parts of speech cause sentence aborts and the processing continues after scanning to the end of the current sentence.

3. Parser

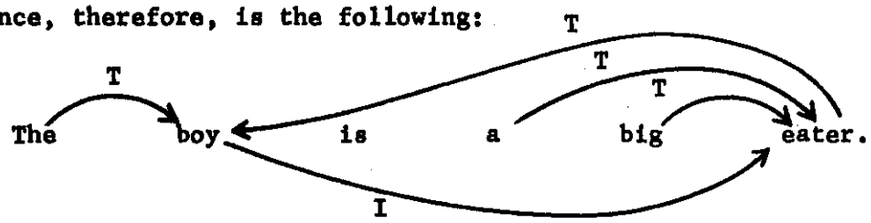
The parser used in DISEMINER is basically that of Klein's automatic essay paraphraser, with a few small modifications [5]. The program is a syntax-directed parser that uses part of speech hierarchy classes and subscripts within each such class to govern the choice of applicable rule. Its parsing stack conveniently embodies the dependencies at the end of a successful parse, with the transitivity or intransitivity governed by the appropriate marker in the stem-part of speech dictionary. Questions to the system are also parsed by this routine, but of course do not have their dependency structure outputted to the dependency matrices. Rather, a dependency pair list is returned to the calling routine which checks M for the presence of all indicated links. Provision is also made for "riddle questions" of the form "What eats, sleeps, and breathes?", in which case a special convention is used in the output dependency list to indicate that a generative search is to be done rather than a simple true-false conclusion.

A special case is provided in the parser for sentences having as their principal verb a form of the stem "to be". The dependency parse for the sentence "The boy is a big eater." on the basis of the inputted grammar would be:



This has several unfortunate features, including the linkage to the very common verb "to be" that is likely to saturate the dependency structure, and also the simply one-way dependency linkage which belies the notion

of restricted equivalence of subject and object. The modified parse of the sentence, therefore, is the following:



This indicates crudely that all boys are likely to be big eaters but not all big eaters are likely to be boys (although some are).

In the case of syntactically incorrect sentences the parser performs what rewrites it can and then summarizes the parsing stack and deduced dependencies, although the sentence is subsequently ignored in the functioning of the system. To enhance the power of the system, some means of multi-path analysis would be desirable, with perhaps multiple initial part of speech tags for the stems as well.

Two system dump procedures are provided, and are called at the end of any successfully completed run as well as in the event of any error. They are designed to print the dependency table and stem dictionary in alphameric form. Since these two tables in a sense comprise the "state" of the system at any moment, they are of great aid in debugging.

4. Compute dependency tables

The implications of all links in the transitive and intransitive dependency matrices are computed and the result is stored in the special dependency matrix 'M'.

5. Dump dependency matrices

The current status of the transitive, intransitive, and 'M' dependency matrices is printed, and a listing of system dictionary (lexical table) is supplied.

6. Input question sentence

This routine is identical to the routine input text sentence. The format for questions is also identical to that of text sentences.

7. All dependencies true

Two possibilities exist for questions. The first is the normal mode of operation and simply involves a testing of all links that are computed during the parse of the question. If all links in the dependency pairs table are found to be true in the 'M' dependency matrix then the question is answered positively. If at least one of the links has not been set in the 'M' matrix then a negative answer is given and a list of all links not found is provided. The other possibility for questions are riddle questions. Formally riddle questions are currently of the form:

$$\text{who/who/N } X_{(1)}\ X_{(2)}\ X_{(3)}\ \cdots\ X_{(k)}\ .$$

where $X_{(i)}$ ::= Y/Z/V | Y/Z/Adj | Y/Z/Prep | W/be/V Y/Z/N | W/be/V Y/Z/Adj | Y/Z/Conj . $i = 1, 2, \ldots k$

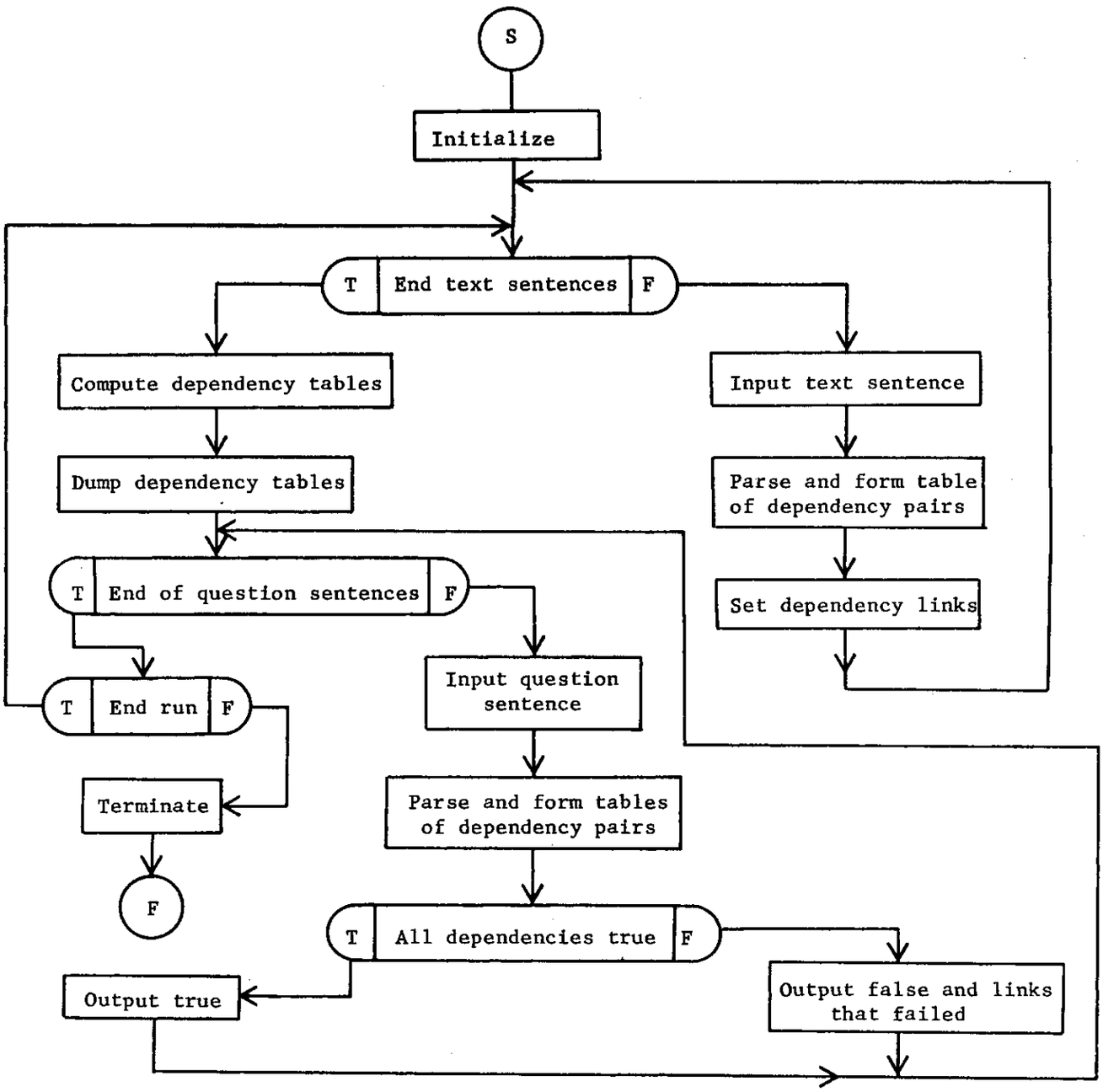
Y and Z are arbitrary alphameric character strings. W is a form of the verb 'to be'. V represents 'Verb'. N represents 'Noun'. Forms of 'to be' and conjunctions (Conj) are ignored and the intersection of the sets $S_1\ S_2\ \cdots\ S_k$ is computed where $S_i = \{T \mid \text{link from stem}(X_{(i)}) \text{ to stem } (T)\}$ when $X_{(i)}$ is not a conjunction or a form of the verb 'to be' and $S_i = \{T \mid T \text{ is a stem}\}$ if $X_{(i)}$ is a conjunction or a form of the verb 'to be'.

Essentially the riddle answerer provides class membership information about the nouns in the system.

If the intersection described above is not empty then a listing

of the stems in the class is provided. If the intersection is empty then the message 'intersection empty' is supplied.

Flow Chart for DISEMINER



3.3 Program performance

The program was written in ALGOL 20 for the CIT G-21. It was run successfully on the G-21 in 32K mode. The compile time of the entire system is approximately 1 minute and 30 seconds. The program proper used 8894 machine words of memory, and 15000 words of memory are available for data. Without resorting to disc tape operations, it is possible to have approximately 400 stems in the system. This, of course, allows a much larger number of words in the system. The time from the beginning of input of sentence until the sentence is parsed and the links set in the dependency matrices takes approximately 3 1/2 seconds per sentence of approximately eight words. Of this time, input takes 1 second, parsing 2 1/2 seconds. The closure of the dependency matrices takes approximately 50 seconds for 50 stems.

4.0 Testing the system

The very simple grammar consisted of the following rules:

$$N_3 \Rightarrow \text{Art}_0 \rightarrow N_2$$

$$N_2 \Rightarrow \text{Adj}_0 \rightarrow N_2$$

$$N_1 \Rightarrow N_1 \leftarrow \text{Mod}_1$$

$$V_2 \Rightarrow V_1 \leftarrow \text{Mod}_1$$

$$V_2 \Rightarrow V_1 \leftarrow N_3$$

$$\text{Mod}_1 \Rightarrow \text{Prep}_0 \leftarrow N_3$$

$$S_1 \Rightarrow N_3 \leftarrow V_3$$

The small arrows indicate the direction of dependency within each construction. As mentioned earlier, links to nouns were transitive; links to forms of the verb 'to be' were recomputed as links to the noun governing that verb.

Also, where text sentences involved constructions beyond the scope of the grammar, grammar codes were assigned to force the sentence to fit the grammar, e.g. in 'The man is mortal', 'mortal' was tagged as a noun. The subscripts determine the order of applicability of the rules in parsing. A complete description of the program that uses rules of this format to perform, simultaneously, a phrase structure and dependency analysis is to be found in the work of Klein [5].

Input text was supplied to the system in two parts. The first test consisted of the following sentences:

See page 17

Input Text I

*1. The is A.

*2. A is the.

3. The girl is a woman.

4. The woman lives in a city.

5. The woman sat on a chair in the house.

6. The girl ate potatoes.

7. The girl owned a dog.

8. LA is a city in California

9. The girl saw an airplane in the air.

10. The woman wrote a letter at a table.

11. The airplane flew to LA.

12. The girl flies to Paris.

13. The man on the ground saw the flying bird.

14. John is a man.

15. John writes a book.

16. John eats dinner at a restaurant.

17. The restaurant near the car is good.

18. The book near completion is bad.

19. The horse eats hay in a stable.

20. The man rides about the park on his horse.

21. The man thinks about the horse.

22. The girl thinks about the man.

---

* Sentences 1 and 2 are added solely as a device to permit the treatment of 'the' and 'a' as equivalents. This information might have been supplied, alternatively, by classifying 'a' as belonging to the stem 'the'. This last device can permit the addition of synonyms to the system.

The table of transitive and intransitive links before merger and closure is contained in appendices 1 and 2, respectively. Plausibility questions based on the first 22 input sentences consisted of a repeat of the input text as questions (for validation only and not presented here) and the following:

Plausibility Questions on Basis of Input Sentences 1-22

1. LA flew to the airplane?

   NOT POSSIBLE: 'fly' not dependent on 'LA'
   'airplane' not dependent on 'to'

2. The chair is in LA?

   NOT POSSIBLE: 'LA' not dependent on 'in'

3. The restaurant is near the girl?

   NOT POSSIBLE: 'girl' not dependent on 'near'

4. The girl lives in a city?

   POSSIBLE

5. The girl sits?

   POSSIBLE

6. The woman flies to LA?

   POSSIBLE

7. The restaurant near completion is good?

   POSSIBLE

8. The book near the car is bad?

   POSSIBLE

9. The man thought about the man?

   POSSIBLE

10. The woman thinks about parks?

    POSSIBLE

After answering these questions, the DISEMINER system processed an additional 10 sentences of text. The transitive and intransitive links were added to those already tabulated and a new closure was computed. Appendix 3 contains the merged links after closure as computed from the entire 32 input sentences. The additional text consisted of the following:

Input Text 2

23. The boy is near the girl.
24. The woman in the park sings songs.
25. The large man in the house smiles at the boy.
26. Men are animals.
27. The girl threw the ball at the wall.
28. John saw sites in LA.
29. Horses are animals.
30. Horses are mortal.
31. The man ate with a fork.
32. The woman eats with a spoon.

Additional plausibility questions were posed to the system after the addition of new text. Note that questions 11 and 18 are repeats of questions posed earlier. At the first query, the system declared them "NOT POSSIBLE". Sentences 23 and 28 of the second input text provided sufficient new information to permit answers of "POSSIBLE" during the second answering period. The new questions consisted of the following:

Plausibility Questions and Answers on the Basis of Sentences 1-32

11(3). The restaurant is near the girl?

POSSIBLE

12. The girl in the park sat on the horse?

POSSIBLE

13. The animal smiled at the wall?

POSSIBLE

14. The smiling man eats dinner?

POSSIBLE

15. The ball threw the girl?

NOT POSSIBLE: 'throw' not dependent on 'ball'
'girl' not dependent on 'throw'

16. Walls smile?

NOT POSSIBLE: 'smile' not dependent on 'walls'

17. Chairs fly?

NOT POSSIBLE: 'fly' not dependent on 'chair'

18(2). The chair is in LA?

POSSIBLE

19. The man eats with a spoon?

POSSIBLE

20. The woman ate with a fork?

POSSIBLE

21. Men are horses?

NOT POSSIBLE: 'horse' not dependent on 'man'

22. Horses are men?

NOT POSSIBLE: 'man' not dependent on 'horse'

Plausibility Questions and Answers on the Basis of Sentences 1-31 (cont'd)

23. Animals are mortal?

POSSIBLE

24. Men are mortal?

NOT POSSIBLE: 'mortal' not dependent on 'man'

25. The horse eats hay in a city?

POSSIBLE

26. The book eats dinner?

NOT POSSIBLE: 'eat' not dependent on 'book'

27. Dinner ate John?

NOT POSSIBLE: 'eat' not dependent on 'dinner'
'John' not dependent on 'eat'

28. John eats California?

NOT POSSIBLE: 'California' not dependent on 'eat'

29. John eats dinner at a dog?

NOT POSSIBLE: 'dog' not dependent on 'at'

The system was then asked a series of riddles. The parser was not used in analyzing the queries -- rather a program mechanically tabulated the intersection of the noun dependencies of the vocabulary items following 'Who' and 'What' ('and' and 'is' excepted). Thus, in riddle three, the system computed the intersection of the lists associated with 'eats', 'potatoes' and 'chairs' as indicated in appendix 3, the table of merged and closed dependency links. The query capability can readily be expanded to cover a wide range of Boolean questions including those

involving negation. The currently existing limitations were necessary because of a time limit placed on the completion of this pilot version of the DISEMINER system. The riddles, based on text sentences 1 to 32 were the following:

Riddles Based on Text Sentences 1-32

1. What flies?

   girl woman airplane bird

2. Who flies and writes?

   girl woman

3. Who eats potatoes and chairs?

   INTERSECTION EMPTY

4. Who eats and is man?

   man John animal

5. Who sees and flies?

   girl woman

6. Who lives and flies and eats?

   girl woman

7. Who thinks and sings and owns?

   girl woman

8. Who thinks rides eats and sees?

   man John animal

9. Who is an animal and thinks?

   man John animal

5.0 Discussion

In one respect, the use of the DISEMINER system to compute semantic classes in terms of dependency criteria that are weakly equivalent to phrase structure criteria involves circular logic. Theoretically, the validity of phrase structure grammar rests upon a hypothetical distributional analysis of all n-tuples of units in terms of concurrence with all other n-tuples [6, 7]. Such an analysis would contain the information one might hope to collect with DISEMINER, and would raise the issue of whether or not ultra refined syntactic classes are not also semantic classes. In reality, no one has ever performed a complete distributional analysis, nor produced a complete phrase structure grammar for any language, nor is it likely that such analysis is possible. The DISEMINER system can be used to collect data for extending the scope of an existing phrase structure grammar. If one wishes to work without a semantic stratum, one may indeed describe the special semantic classes as syntactic classes; or if one would work with a stratified system, collect data for semantic units that map onto another level or stratum. It is because of the absence of a complete phrase structure grammar that the methodology of DISEMINER does not in involve circularity.

5.1 Semantic classes and saturation

An examination of the merged and closed links table (appendix 3) indicates that the dependency links of function words (prepositions, articles, etc.) will quickly become saturated, that is, as more and more text is analyzed, these lists will come to include nearly the entire vocabulary of the language. On the other hand, the link lists pertaining

to nouns and verbs appear to be the most likely to yield interesting semantic classes. Boolean queries of the type used in the riddles (as well as more powerful types) can be expected to obtain semantic classes at any level of refinement, including, at the extreme, classes consisting of one element each.

5.2 Refinements and extensions of the DISEMINER system

1. The English grammar used in the testing of the system was extremely limited. Grammars of vastly increased complexity are essential to a full scale use of the system. The parser used with the system did not handle ambiguity; rather, it produced a unique analysis for each sentence. The program design readily permits the substitution of other parsers. Interestingly, the introduction of multiple analyses of sentences need not interfere with the computation of distribution classes -- it would seem a legitimate course to plot the dependencies implied by all valid analyses.

2. It is worth mentioning that the plausibility question answering mechanism can also be used as a two-level parsing system that checks first for syntactic, then semantic well-formedness.

3. As a general information retrieval system in its own right, DISEMINER is not suitable because of saturation. It is conceivable that a DISEMINER might be useful for retrieval of information in a highly specialized subject area. A system containing a family of DISEMINERS, each storing a semantic map of a particular subject area, might avoid the problem of saturation and, as a whole, function with a broad data base.

4. The testing of the system provides evidence for the hypothesis

that dependency notation, although weakly equivalent to phrase structure notation, is perhaps a more powerful device for computation purposes. Graph theory is a well developed mathematical area and the theorems of that field are available for any computation in linguistics involving dependency networks. In building models with directed graphs, one is not limited just to the relations of 'transitive' and 'intransitive'. One may label links with any number of relations deemed pertinent to an analysis. The use of such descriptions (which can also be called directed graphs with colored edges -- 'transitive' and 'intransitive' would be two such colors) might greatly facilitate the formalization of the relevant linguistic theory. This remark is intended to apply to transformational grammars as well as other kinds.

5. The testing of the DISEMINER system has been extremely limited. All conclusions pertaining to its usefulness as a research tool are tentative. Nevertheless, the results that have been obtained do suggest that further testing and improvement of the system is warranted.

REFERENCES

[1] Lamb, S., The Sememic Approach to Structural Semantics. In Kimbel A. Romney and Roy D'Andrede (Eds.), Transcultural Studies in Cognition. American Anthropologist, 1964, 66(3), Part 2.

[2] Simmons, R. F., Klein, S., and McConlogue, K., Indexing and Dependency Logic for Answering English Questions. American Documentation, Vol. 14, No. 3, July 1964.

[3] Gaifman, H., Dependency Systems and Phrase Structure Systems. Memorandum P-2315, The RAND Corporation, Santa Monica, California, 1961.

[4] Hays, D. G., Dependency Theory: A formalism and some observations. Language, 1964, 40(4), 511-525.

[5] Klein, S., Automatic Paraphrasing in Essay Format. Mechanical Translation, Vol. 8, Issues 3 & 4 combined, August-December, 1965.

[6] Wells, R. S., Immediate Constituents. Language, 23, 1947, 81-117.

[7] Harris, Z. S., Methods in Structural Linguistics. University of Chicago Press, Chicago, 1951.

[8] Harris, Z. S., Distributional Structure. Word, Vol. 10, No. 2-3, August-December, 1954. 146-162.

[9] Warshall, S., A Theorem on Boolean Matrices. Journal of the ACM, Vol. 9, No. 1, January 1962, 11-12.

Appendix 1

Transistive Links Before Merger and Closure After Sentence 1-22

1.	the	a	girl	woman	house
		airplane	air	man	ground
		bird	book	restaurant	car
2.	be				
3.	a	the	woman	city	chair
		dog	airplane	letter	table
		man	book	restaurant	stable
4.	what				
5.	who				
6.	girl				
7.	woman	girl			
8.	live	woman			
9.	in	city	chair	airplane	hay
10.	city	LA			
11.	sit	woman			
12.	on	man	park		
13.	chair				
14.	house				
15.	eat	girl	John	horse	
16.	potato				
17.	own	girl			
18.	dog				
19.	LA				

20.	California			
21.	see	girl	man	
22.	airplane			
23.	air			
24.	write	woman	John	
25.	letter			
26.	at	letter	dinner	
27.	table			
28.	fly	girl	airplane	bird
29.	to			
30.	Paris			
31.	man	John		
32.	ground			
33.	bird			
34.	John			
35.	book			
36.	dinner			
37.	restaurant			
38.	near	book	restaurant	
39.	car			
40.	good	restaurant		
41.	complete			
42.	bad	book		
43.	horse			
44.	hay			
45.	stable			
46.	ride	man		

47. about

48. park

49. his horse

50. think girl man

Appendix 2

Intransitive Links Before Merger and Closure After Sentences 1-22

1.	the	a	
2.	be		
3.	a	the	
4.	what		
5.	who		
6.	girl	woman	
7.	woman		
8.	live		
9.	in	live	
10.	city	in	
11.	sit		
12.	on	sit	
13.	chair	on	
14.	house	in	
15.	eat		
16.	potato	eat	
17.	own		
18.	dog	own	
19.	LA	city	to
20.	California	in	
21.	see		
22.	airplane	see	
23.	air	in	

24.	write		
25.	letter	write	
26.	at		
27.	table	at	
28.	fly		
29.	to	fly	
30.	Paris	to	
31.	man	about	
32.	ground	on	
33.	bird	see	
34.	John	man	
35.	book	write	bad
36.	dinner	eat	
37.	restaurant	at	good
38.	near		
39.	car	near	
40.	good		
41.	complete	near	
42.	bad		
43.	horse	on	about
44.	hay	eat	
45.	stable	in	
46.	ride		
47.	about	ride	think
48.	park	about	
49.	his		
50.	think		

Appendix 3

Merged Links with Closure after Sentences 1-32

1.	the	the	a	girl	woman
		in	city	on	chair
		house	own	dog	LA
		see	airplane	air	write
		letter	at	table	to
		man	ground	bird	John
		book	restaurant	near	car
		good	bad	horse	stable
		about	park	boy	animal
		throw	ball	wall	mortal
		with	fork	spoon	
2.	be				
3.	a	the	a	girl	woman
		in	city	on	chair
		house	own	dog	LA
		see	airplane	air	write
		letter	at	table	to
		man	ground	bird	John
		book	restaurant	near	car
		good	bad	horse	stable
		about	park	boy	animal
		throw	ball	wall	mortal
		with	fork	spoon	

4.	what				
5.	who				
6.	girl	woman	near		
7.	woman	girl	woman	near	
8.	live	girl	woman	near	
9.	in	girl	woman	live	in
		city	on	chair	eat
		LA	see	airplane	to
		man	John	near	hay
		about	animal	site	
10.	city	in	city	LA	to
11.	sit	girl	woman	near	
12.	on	in	sit	man	John
		about	park	animal	
13.	chair	on			
14.	house				
15.	eat	girl	woman	on	man
		John	near	horse	about
		animal	mortal		
16.	potato	eat			
17.	own	girl	woman	near	
18.	dog	own			
19.	LA	in	city	to	
20.	California	in			
21.	see	girl	woman	man	John
		near	about	animal	
22.	airplane	see			

23.	air	in			
24.	write	girl	woman	man	John
		near			
25.	letter	write			
26.	at	eat	write	letter	dinner
		smile	throw	ball	
27.	table	at			
28.	fly	girl	woman	see	airplane
		bird	near		
29.	to	fly			
30.	Paris	to			
31.	man	man	John	about	animal
32.	ground	on			
33.	bird	see			
34.	John				
35.	book	write	bad		
36.	dinner	eat			
37.	restaurant	at	good		
38.	near	write	at	book	restaurant
		near	good	bad	boy
39.	car	near			
40.	good	at	restaurant	good	
41.	complete	near			
42.	bad	write	book	bad	
43.	horse	on	about	animal	mortal
44.	hay	eat			
45.	stable	in			

46. ride	man	John	about	animal
47. about	ride	think		
48. park	in	about		
49. his	on	horse	about	animal
	mortal			
50. think	girl	woman	man	John
	near	about	animal	
51. boy	at	near		
52. sing	girl	woman	near	
53. song	sing			
54. large	man	John	about	animal
55. smile	man	John	about	animal
56. animal	on	man	John	horse
	about	animal	mortal	
57. throw	girl	woman	near	
58. ball	throw			
59. wall	at			
60. site	see			
61. mortal	on	horse	about	animal
	mortal			
62. with	eat			
63. fork	with			
64. spoon	with			
65. walls				
66. and				