3

FROM WORDS TO NUMBERS: A SET THEORY FRAMEWORK FOR THE COLLECTION, ORGANIZATION, AND ANALYSIS OF NARRATIVE DATA

Roberto Franzosi[*]

Recent advances in computer-assisted content analysis have made available to historians and social scientists much richer and more flexible historical records from textual sources (e.g., archives, newspapers). These records are organized in complex data structures (namely, separate but interconnected files). Furthermore, the content of these records is mostly words. This paper shows how narrative data organized in complex data structures can be analyzed statistically. It also shows that powerful linguistic schemes for content analysis (namely, semantic grammars) can be reexpressed as set theoretical problems. Set theory also provides the mathematical foundation for the relational data model that can be used to store text data collected via a semantic grammar. Finally, set theory provides the basic tools (namely, the cardinal number) necessary to go "from words to numbers." It is this basic transformation that allows researchers to perform general kinds of quantitative analyses on such qualitative data as words.

This paper was written partly with the financial assistance of Stoà-Formez. The paper is part of a larger project on industrial conflict in Italy which has received support from the Graduate School of the University of Wisconsin, The National Science Foundation FORMEZ (Rome, Italy), Intersind (Rome, Italy), Confindustria (Rome, Italy), and Consiglio Nazionale delle Ricerche. I would like to thank Svetlana Kropp for her help with the empirical work presented in the paper.

*Rutgers University

1. INTRODUCTION

Semantic text grammars provide a novel and powerful tool for collecting textual data (Franzosi 1989). A grammar organizes textual data around the simple structure Subject/Action/Object (the "canonical form" of the language) and their modifiers (e.g., time, space). Although semantic grammars work well only for narrative, narrative provides a large set of texts of interest to social scientists.[1]

The use of text grammars in a computer environment as data collection schemes yields highly flexible and rich data. The very richness and flexibility of the data collected via a semantic grammar, however, may seem to pose several problems. First, how do we organize this mass of data? Second, how do we analyze it? Traditionally, we are used to thinking of data for statistical analysis as a single rectangular array where each column represents a variable and each row a case (or vice versa). Each element of this array contains a number. None of this is true when using a text grammar. Although the data are organized in rectangular arrays, there are typically distinct arrays for different types of information (e.g., an array for subjects, another one for actions, etc.). Furthermore, these distinct arrays are interconnected in complex ways (e.g., the array of subjects to that of actions, the arrays of subjects and actions to those of their modifiers). Finally, the elements of each of these arrays are mostly words. How can we compute averages, standard deviations, run regressions, and statistical tests on words?

Available methodologies allow us to perform statistical analyses of text data using words directly as input (Benzecri 1980; Lebart and Morineau 1984). These methodologies, however, have only limited statistical capabilities (namely, exploratory data analysis). Furthermore, they require extensive manipulation of text input, in order to make the text suitable for automatic analysis (see the appendix to this chapter). My concern in this paper is different: How can we analyze text data collected via a semantic grammar using more general statistical techniques (e.g., regression)? To find an answer to this question, I use the theoretical framework of mathematical set theory. Set theory provides the basic concepts that allow us to make the transition from the qualitative to the quantitative. On numbers, we

[1]Narrative, for instance, characterizes newspaper articles (Franzosi 1987), archival documents (Shapiro and Markoff, Forthcoming), and medical records (Sager 1981, pp. 213–31; Sager et al. 1987; Sager 1987).

can apply any kind of statistical technique. Set theory, however, offers a solution not only to the problem of data analysis but also to the second problem, that of data organization. Relational DataBase Management Systems (DBMS), which are powerful tools for organizing complex data structures in a computer environment, have their theoretical foundations in set theory. More generally, I show that even the semantic grammar proposed as a data collection scheme can be reexpressed as a set theoretical problem. Thus the same basic mathematical theory (set theory) provides a general framework for the collection, organization, and analysis of text data. To the extent that a semantic grammar of data collection can be easily implemented in commercially available relational DBMSs, the use of such complex coding schemes is well within the reach of any PC user. Couching the problem of the collection, organization, and analysis of text data in set theoretical terms is thus not just a question of mathematical elegance but also of practical utility.

2. SEMANTIC GRAMMARS

In recent decades, linguists have argued that text is characterized not only by syntactic structures but also by semantic structures (for an introduction to semantic grammars, see Franzosi 1989). Just as a syntax grammar captures the surface representation of text, a semantic grammar captures its deep representation. Semantic grammars provide overall schemata or macrolevel organizational templates for text genres. Thus, for instance, a sociology journal article is typically characterized by the following deep structure: introduction, literature review-theory, data and methodology, empirical results, conclusions. Similarly, news reports follow the typical format: summary (the rule of the five W's: who, what, when, where, why), description of the main and related episodes, causes and consequences, editorial comments. Narrative text exhibits even more microlevel semantic structures in the form Subject/Action/Object (SAO) and their modifiers (e.g., time, space) (Franzosi 1989). The structure is quite general for a whole class of narrative text, from Russian folk tales, to police or newspaper factual accounts of events, to medical reports. Only the labels of the modifiers would change, according to the specific meaning that one wants to extract (e.g., the actor modifier <trade union> in labor research, or <patient's symptoms> in medical research). Basic SAO structures (we could call each individual SAO structure "semantic triplet") can be assembled into higher-level semantic aggregates, such as episodes and stories. Again, the labels of these aggregates would change from one substantive field to another (e.g., subplots and plots in the study of folktale, event and campaign in collective action research). The number of these aggregates would also change in response to a researchers' needs (e.g., doctor's visits, illness, and patient case history in the study of medical records).

To understand how a semantic grammar provides an organizational framework for text data, let me code part of a newspaper article.

> **New Unrest Is Reported in Miami**
> Miami, June 27—Two police officers responding to a gunfight on a street in a racially intense neighborhood shot and wounded a man today, setting off a night of unrest in which about 50 people were arrested, the police said.
>
> Crowds of young blacks took to the streets hurling rocks and bottles. A police officer was hospitalized after he was struck in the head by a stone, but there were no reports of other injuries.
>
> The police said the disturbances spread from the Overtown neighborhood, north of Miami's downtown business district, to the Liberty City section, where shooting was reported. The neighborhoods were the scenes of three days of disturbances in 1989 after a policeman killed a black motorcyclist. . . . (*New York Times,* Mid-West Edition, June 28, 1991, p. A16)

With events and triplets organized in chronological order and passive sentences turned into active form,[2] output reformatted within the categories of a semantic grammar appropriately designed for the study of race disturbances would look something like this:[3]

[2]Transformation of passive sentences to active is made necessary by the SAO structure. I have used the question mark symbol (?) to denote the unknown subject of passive sentences turned into the active form. When the subject can be inferred from context, I put that in parentheses along with the question mark symbol.

[3]Sager (1987, p. 4) calls this type of text coding "information formatting."

<event 1> 1989 Miami race disturbances
<time> 1989
<Semantic Triplet 1> <subject> police officer <number> 1
<action> kills <object> motorcyclist
<number> 1 <race> black
<Semantic Triplet 2> <subject>? (crowds) <action> engage in disturbances <duration> three days
<space> Miami, Overtown neighborhood, north of Miami's downtown business district, and the
Liberty City section

<event 2> 1991 Miami race disturbances
<time> today (June 27)
<Semantic Triplet 1> <subject> police officers <number> 2
<action> respond to gunfight <time> today <space> street, racially intense neighborhood
<Semantic Triplet 2> <subject> police officers <number> 2
<action> shoot <object> man <number> 1
<Semantic Triplet 3> <subject> police officers <number> 2
<action> wound <object> man <number> 1
<Semantic Triplet 4> <subject> crowds <type> young <race> black <action> take to the streets
<Semantic Triplet 4> <subject> ? (crowds) <action> spread disturbances <space> Miami, from the Overtown neighborhood to the Liberty City section
<Semantic Triplet 6> <subject> crowds <action> hurl rocks
<Semantic Triplet 7> <subject> crowds <action> hurl bottles
<Semantic Triplet 8> <subject> ? (crowds) <action> strike
<instrument> stone <object> police officer
<number> 1 <body part> head
<Semantic Triplet 9> <subject> ? (ambulance) <action> take to hospital <object> police officer
<Semantic Triplet 10> <subject> people <number> no other
<action> suffer injuries
<Semantic Triplet 11> <subject> ? (police) <action> arrest
<object> people <number> about 50

As the example shows, a semantic grammar constitutes a powerful "coding scheme," with a series of advantages over more traditional "thematic" content analysis and even more recent computer applications (Franzosi 1989, 1990*a*, 1990*b;* see the appendix to this chapter). A grammar preserves the interconnections between various elements of discourse (e.g., actors are linked to actions, and both actors and actions are linked to their characteristics); a grammar produces coded output with much of the narrative flavor of the original text and higher data reliability.

Despite these advantages, semantic grammars are not without problems. First, the simple linguistic structure of a semantic grammar, namely Subject/Action/Object and modifiers, is not well suited for organizing nonnarrative text (e.g., a scientific paper, a philosophical treatise, a newspaper editorial). Second, although the application of a semantic grammar to narrative text minimizes the semantic problems involved in the coding process, it by no means solves those problems (problems better known in content analysis as intercoder reliability).

3. SET THEORY: A REVIEW OF USEFUL CONCEPTS

In this section I review some basic concepts of set theory: set, ordered pairs and product sets, sentences and relations, and cardinal numbers (for an easy treatment of the concepts discussed in this section, see Halmos 1960). I show how set theory can subsume all the basic concepts of a text grammar. Set theory, however, also contains concepts that are not part of the linguistic conceptual apparatus but are indispensable for a solution to our problem of organizing and analyzing text data.

3.1. *Set*

In set theory, the concept of set is a metaconcept—that is, it is not formally defined. Intuitively, though, a set is a collection, list, or class of objects. Thus, in a notation called the *tabular form* of a set, S = {police officer, motorcyclist, ? (crowds) . . .} indicates the set of the actors involved in the newspaper article on the Miami disturbances. Alternatively, in the *set-builder form* $S = \{s \mid s \text{ is noun}\}$ which reads "S is the set of actors s such that (the symbol | reads "such that") s is an actor." The notations $s \in S$ and $s \notin S$ read respectively

"s belongs to S" and "s does not belong to S." The elements of a set must be distinct and unique. Needless to say, a semantic grammar can be expressed conveniently in set theoretical notation, with a set R for actoRs (subjects or objects), a set N for actioNs, a set M for Modifiers, etc.

3.2. *Ordered Pairs and Product Sets (Cartesian Products)*

The concepts of ordered pair, Cartesian product, and relation allow us to link specific entries in one set to specific entries in another set. An *ordered pair* consists of two elements, say a and b, where a is the first element and b the second. An ordered pair is denoted by (a, b), for instance, (state appeals court, officer). If A and B are any two sets, the *product set* or *Cartesian product* of A and B consists of all ordered pairs (a,b) where a $\in A$ and $b \in B$. It is denoted by $A \times B$ (reads "A cross B"). Using the set-builder notation, this is $A \times B = \{(a,b) \mid a \in A, b \in B\}$. Thus, if S is a set of actors such as S = {police officer, motorcyclist, ? (crowds), . . .} and A is the set of actions A = {kills, engage in disturbances, . . .}, the Cartesian product of S and A consists of all ordered pairs (police officer, kills) (police officer, engage in disturbances) . . . (police officer, hurl rocks). . . . More generally, the Cartesian product of n sets $S_1, S_2, \ldots, S_n$ is denoted by $S_1 \times S_2 \times S_3 \times \ldots S_n$ and consists of all n-tuples $(s_1, s_2, s_3, \ldots, s_n)$ where $s_1 \in S_1, s_2 \in S_2, \ldots, s_n \in S_n$. Thus the Cartesian product $S \times A \times O$, where S and A are the sets of actors and actions defined above, and S is equal to O (i.e., both Subjects and Objects are actors), gives the following ordered 3-tuples (police officer, kills, motorcyclist) . . . (police officers, hurl bottles, people). . . .

3.3. *Sentences and Relations*

To competent language users not all of the 3-tuples that result from the Cartesian product $S \times A \times O$ will sound equally semantically well formed. Thus the 3-tuples (state appeals court, kills, crowd) and (policeman, hurl bottles, motorcyclist) do not square away with the reader's world knowledge. Clearly, we need to be able to restrict to semantically acceptable n-tuples the combinations that result from set joins. The concepts of *sentence* and *relation* allow us to do just that.

Given the Cartesian product $A \times B$ of two sets A and B, we can define as *propositional function* or *sentence* the expression $P(a,b)$. This expression has the property that $P(a,b)$ is either true or false for any ordered pair $(a,b) \in A \times B$. The *relation* $R = (A,B,P(a,b))$ is the Cartesian product of sets A and B subject to the constraint of the sentence $P(a,b)$. If $R = (A,B,P(a,b))$ is a relation, the solution set R^* of the relation R consists of the elements (a,b) in the Cartesian product $A \times B$ for which $P(a,b)$ is true. Notationally, $R^* = \{(a,b) \mid a \in A, b \in B, P(a,b) \text{ is true}\}$. Thus the set of semantic triplets of the grammar is the subset of acceptable 3-tuples obtained from the Cartesian product of the sets of actors and actions $S \times A \times O$—that is, $T = \{S \cup A \cup O \mid (s,a,o) \notin C, s \in S, a \in A\}$ where $\cup$ is the set symbol for a *union* or sum of sets (in this case, of the sets S, A, and O) and C is the set of "selection restrictions"—that is, the set of invalid 3-tuples.

3.4 *Cardinal Numbers*

The set theoretical concepts of ordered pairs, Cartesian products, sentences, and relations have allowed me to reexpress a semantic grammar in a tight mathematical notation. These concepts, however, have not allowed me to transform words into numbers. Thus far I have remained in the domain of words. It is the *cardinal number* that allows me to make the transition. The cardinal number of each of the following sets {crowd} {hurl rocks, hurl bottles} is, respectively, 1 and 2. The cardinal number, in other words, measures the size of a set: It gives the number of elements in a set.[4] It is the cardinal number that allow us to reexpress a list of words in numeric terms.

In conclusion, set theory offers a simple but powerful mathematical framework for all the linguistic concepts of a text grammar. The advantages of using a mathematical rather than a linguistic framework are twofold. First, set theory provides a simple mechanism to go "from words to numbers": the cardinal number. When we move from words to numbers, we move into the more familiar terrain of statistical data analysis. Second, set theory constitutes the mathematical foundation of Relational DBMS methodology. It is

[4]I provide only an intuitive understanding of cardinal numbers and cardinality, particularly as they refer to finite sets. I purposely avoid a formal definition of cardinal numbers and cardinality (see, however, Halmos 1960).

this methodology that makes feasible the use of such complex coding schemes as semantic grammars in a computer environment.

4. COMPLEX DATA STRUCTURES

In principle, it is possible to use a semantic grammar in a pencil and paper environment (on these issues, see Franzosi 1990*a*). However, for large scale data collection projects this solution is impractical. A grammar of data collection is generally based on a large number of coding categories, on a complex network of relations between categories, and on a large dictionary of precoded items. A grammar of data collection can also yield large volumes of coded material. The use of traditional content analysis tools (pencil and printed coding schemes) is incompatible with large volumes of highly complex data. The application of a text grammar in large scale projects requires a computer environment, where dictionary and coded material can be easily stored and retrieved. (For a computer application of a semantic grammar for collecting event data, see Franzosi 1990*b*.)

The implementation of a text grammar in a computer environment eliminates some problems (for one thing, it makes feasible the use of such complex coding schemes as semantic grammars), but it also creates new ones. In particular, how can we physically organize the data collected via a semantic grammar? How can we preserve the complex network of relations typical of a text grammar?

Data in a computer are stored in files. Files are collections of records. Records contain one or more fields where different types of information are stored (Ullman 1980, p. 20). Typically, one record contains all the fields necessary to store all available information (rectangular data structure). This type of data organization, however, breaks down when the information to be stored has a complex structure. Imagine, for instance, designing the record layout for a semantic grammar. We need a field for actor, one for all its modifiers, a field for action, another for its modifiers. What do we do in case we have more than one actor, or the same type of modifier appears more than once? Should we leave room in the record for more than one occurrence of each field? How many occurrences of a field should we allow for? Two? Three? Five? Ten? The higher the number, the more likely it is that we will have enough room to store repeated information on the record. The higher the number, however, the higher is also the likelihood of wasted space, whenever there are zero or fewer occurrences than the maximum limit allowed (Ullman 1980, pp. 55–56). In any case, what do we do when the number of repetitions exceeds the maximum limit?[5]

Clearly, a simple, rectangular data organization (of the type one record and one file) is inadequate for problems characterized by complex information structures. For these cases, we need to organize information in separate and interconnected files. Together, this set of interconnected files constitutes a database (Ullman 1980, p. 1).[6] Databases offer a more efficient solution for the storage of complex data structures. In addition, they offer several other advantages: fewer redundancies and inconsistencies, data sharing, centralized enforcement of standards and security, and maintenance of data integrity (Date 1981, pp. 9–12).

Despite these advantages, social scientists have been very slow in moving away from traditional, rectangular forms of data organization and into more complex forms of data structures; see, however, the panel data Panel Study of Income Dynamics (PSID) in Solenberger, Servais, and Duncan (1989) and Survey of Income and Program Participation (SIPP) in David (1989), or the cross-sectional data National Study of Families and Households in Winsborough (1989). There are several reasons for the limited use of DBMS technology in the social sciences.

One of the reasons is that social scientists collect data in order to analyze them statistically. On the one hand, most statistical packages expect rectangular data arrays as input. On the other hand, DBMSs only allow the most basic statistical functions (namely, frequency distributions). In order to perform more complex statistical analyses of data stored in database form, an intermediate step is necessary: the data required for statistical analysis must be first written out in a rectangular file (Doyle 1989, p. 185). In any case, general purpose

[5]This type of record structure with repeating groups is called "variable length record format." (Ullman 1980, pp. 52–58; Date 1981, pp. 41–42). There are several strategies available to store information in these more complex structures.

[6]There are several powerful, commercially available software packages for the management of the files of a database. This software is called DataBase Management System (DBMS) (Ullman 1980, p. 1). A DBMS allows users to enter, edit, and retrieve data "in abstract terms, rather than as the computer stores the data" (Ullman 1980, p. 1).

DBMSs are less efficient than conventional, sequential access programs when all the data, rather than a few selected records, need to be retrieved (Doyle 1989, pp. 183–84). For purposes of data analysis, all records are typically retrieved. Furthermore, working in a DBMS environment and on a complex data structure presupposes a very good knowledge of both the data structure and relational theory; it also requires a full-time data manager (Doyle 1989, p. 185).

4.1. *Data Models*

All databases share some basic concepts, such as "entities," "entity sets," and "relationships." According to Ullman (1980, p. 11), an "entity is a thing that is distinguishable; that is we can tell one thing from another"; see also Date 1981, p. 8. In our grammar, actions are entities since a "competent reader of the language" can distinguish them, for instance, from actors. Entities can have characteristics that are called attributes. Space, Firm, Actor Type, Number of Actors, the Actor modifiers of our grammars could be attributes of the entity "actor." Groups of similar entities form entity sets, such as actor set, action set, etc. Space, Time, Action Type, Number of Actions, Instruments—the Action modifiers—could be attributes of the entity set "action." Each entity in an entity set has a domain of values for a given attribute. Thus the entity "strike" that belongs to the action entity set, may have the following domain in the attribute Action Type: wildcat, sit-in, revolving, work-to-the-rule, general. The entity "charge" of the action entity set, may have the following domain in the attribute Instrument: baton, water cannon, fire-arms. Typically, in a database, each entity set is stored in a separate file. The information contained in the newspaper article presented above (the Miami race disturbances) could be organized in separate files (e.g., event, triplet, actor, action, time), each containing the relevant textual information supplied by the newspaper article.

With the information stored in separate files in a database, we need a way to link the various files together. Without these links, for instance, there is no way to know which actors perform which actions, or where and when each action occurs. The concept of relationship refers to the link(s) between (among) the entities of different entity sets (Ullman 1980, p. 13). Relationships can be one-to-one—i.e., for each entity in either set there is at most one entity in the

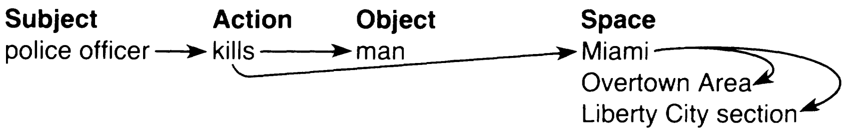

FIGURE 1. Representation of relationships in the hierarchical model.

other set. More common, though, is the many-one relationship, where one entity in entity set S_2 is associated with zero or more entities in set S_1, but each entity in S_1 is associated with at most one entity in S_2 (in this case, the relationship is called many-one form S_1 to S_2). The grammar of data collection is a good example of many-one relationship: one or more actors per semantic triplet, one or more modifiers per actor, etc. In the many-many relationship, there are no restrictions on the set of pairs of entities that may appear in a relationship set.

The different types of relationships correspond to different types of data structures (data models).[7] The most common data models are the hierarchical (many-to-one relationships), the network and the relational models (many-to-many relationships) (Ullman 1980, Date 1981). Data models basically differ in the way they deal with relationships among the entities of different entity sets (i.e., among data in various files) (Date 1981, p. 78).

In the hierarchical and network models, relationships are represented via pointers, where a pointer is an integer value that corresponds to the physical location of a record in a file where a particular piece of data is stored. In a graphical representation, pointers are typically indicated by arrows. The relationship between the entities of the Actor, Action and Space Entity Sets of the Miami disturbances story is represented in Figure 1.

In the relational model relationships between files (or tables, as they are called in relational terminology) are represented by at least one common field in two different files (Date 1981, p. 65). Thus, for instance, the relationship between triplets, actors, actions,

[7]All data models consist of two elements: (1) a mathematical framework and notation for expressing data and relationships and (2) operations that express queries and other manipulation of the data (data entry and editing) (Ullman 1980, p. 18).

and space for the Miami disturbances story could be represented as shown in Figure 2.[8]

The triplet_table has four columns (also called attributes; the rows are also called tuples): the triplet, subject, action, and object identifiers. The actor_table (subjects and objects) has three attributes: the actor identifier, a boolean field to indicate whether the entry is a Subject (value = 0) or an Object (value = 1), and the actor name. The action_table has two attributes: the action identifier and the action name. The space_table has four attributes: action identifier, direction (in order to express movement), city, and neighborhood names. The triplet and action tables have in common the attribute action_id. The triplet and actor tables have in common the actor_id (either subject_id and object_id in the triplet table). The action and space tables share the action_id. The overlap of at least one field between different tables is altogether characteristic of relational data models; in fact, it is through overlapping attributes that relationships between tables are established in relational models (Date 1981, p. 65). Thus the same action_id 2, between the space_table and action_table tell us that the Miami, Overtown neighborhood and the Liberty City section are the stage for the action "engage in disturbances" (action_id 2, in the action_table). If we look up action_id 2 in the triplet table we find that the subject_id of the triplet is 3. If we want to find out the name of the actor that performed the action "engage in disturbances," we would then look up in the actor_table the actor_id 3. The result is: ? (crowds), classified as the subject of the action (with a value of zero in the Subject/Object attribute). Thus, in the sample database, the triplet table contains the key to reconstructing the SAO structure and the key to connecting actors (subject and objects) to actions, via the subject_id, object_id, and action_id.

The hierarchical structure of the grammar (semantic triplets under events, just to mention two higher-level aggregates) would seem to translate naturally into a hierarchical data model. However, relational models have several advantages over either the hierarchical or network data models (Ullman 1980, p. 98). These advantages could ultimately make a relational data model a better choice for a

[8]There are a variety of ways of building a relational SAO structure and its modifiers. The example I provide here is a very simple one, for illustrative purposes.

triplet_table

triplet_id	subject_id	action_id	object_id
1	1	1	2
2	3	2	
3	4	3	
4	5	4	6
5	7	5	8
6	9	10	
7	10	11	

actor_table

actor_id	Subject/Object	actor_name
1	0	police officer
2	1	motorcyclist
3	0	? (crowds)
4	0	police officers
5	0	police officers
6	1	man
7	0	police officers
8	1	man
. . .		

action_table

action_id	action_name
1	kills
2	engage in disturbances
3	respond to gunfight
4	shoot
5	wound
6	take to the streets
7	spread disturbances
. . .	

space_table

action_id	direction	city	neighborhood
2		Miami	Overtown
2		Miami	Liberty City section
3		Miami	street, racially intense neighborhood
7	from	Miami	Overtown
7	to	Miami	Liberty City section
. . .			

FIGURE 2. Representation of relationships in the relational model.

computer representation of the grammar, despite the intuitive similarity between a text grammar and a hierarchical data model.

First, the mathematical foundation of the relational model (relational calculus and algebra, or set theory) is easy to grasp. Second, the data organization of relational models is also easier to understand than that of hierarchical or network models: A table is a more familiar concept than a pointer. Third, the hierarchical model works very well for truly hierarchical structures only—that is, structures characterized by one-to-many relationships between entity sets (e.g., many triplets related to one event). For nonhierarchical structures characterized by many-to-many relationships, the model runs into a series of problems. Furthermore, with time, even truly hierarchical structures tend to naturally evolve into more complex structures characterized by many-to-many relationships (Date 1981, p. 69). Even the semantic grammar of data collection, seemingly hierarchical in nature, breaks down in some limiting cases into a typical many-to-many relationship.

To illustrate this problem, let us consider an example: three labor disputes by metal, chemical, and textile workers centered around the issue of the renewal of their collective contract. These three disputes are separate and distinct, since each of the categories of workers involved has specific demands, specific unions, and specific timetables and strategies. The three struggles proceed together in time but independently of each other. Let us assume, however, that at some point, the unions of the three categories of workers decide to call a joint general strike to display workers' solidarity and to exert united pressure on the employers. The problem is to decide under which dispute we code the general strike. Under just one of the three disputes? Clearly, this would misrepresent the unfolding of events. Then, should we code the event under all three disputes? In which case how do we make sure that the same strike is not counted as three different events? In a truly hierarchical model there is no way to establish connections between trees. The branches of one tree (e.g., an event "strike" under the metal workers' dispute) do not touch the branches of another tree (e.g., an event "strike" under the textile workers' dispute). Although technical solutions to these problems are available (namely, cross pointers), the problems do highlight undesirable properties of hierarchical data models. Finally, hierarchies can easily be expressed within a relational model. In a relational model, each hierarchy can be set up as a specific table (or relation). Thus, for instance, we could specify one relation for events and one for collective campaigns.

campaign_table

campaign_id	campaign_name
1	Miami race disturbances

event_table

event_id	campaign_id	event_name
1	1	1989 disturbances
2	1	1991 disturbances

Just adding new tables in a relational model is not sufficient, however. For every new table that we add, we also need to modify all those tables, among the existing ones, that we want to join directly to the new table (after all, joins are based on the existence of at least one overlapping attribute between tables). This requires adding to the existing tables a new column for the identifier of the new table. Although this may seem cumbersome, in fact, the procedure points to one of the great advantages of relational technology. One does not need to start data collection with a fully specified grammar. All the hierarchies, all the modifiers need not be known in advance. One can add complexity to the grammar as necessity calls for and as data collection proceeds, by simply adding new relations (and new columns to existing relations in order to make table joins possible) (see Winsborough 1989, p. 246).

In conclusion, the relational model is more general than the hierarchical or network data models—and hierarchies can easily be expressed within a relational model. It is also more user friendly, being based on one concept—relation—and on a simple but powerful mathematical notation—set theory. Since a semantic grammar of data collection translates quite easily into a set theory notation, a relational data model is an appropriate organizational structure for data collected via a semantic grammar.[9,10]

[9]For these reasons, relational databases have become increasingly popular. Until recently, commercial database systems were also exclusively based on the network or hierarchical model (Ullman 1980, p. 100).

[10]Relational methodology with its typical tale constructs has been successfully applied to store and retrieve such textual material as medical records (Sager et al. 1987; Sager 1981, pp. 213–39) or questionnaire responses such as SIPP (Survey of Income and Program Participation; see Sager et al. 1987; David 1989).

5. RELATIONAL DATABASE MANAGEMENT SYSTEMS AND SQL

There are several commercially available Relational DBMSs. Until recently, each Relational DBMS had its own language and set of commands (e.g., QUEL in INGRES; see Ullman 1980, pp. 141–48). Switching from one DBMS to another required learning an entirely new set of commands. SQL (Structured Query Language), however, is fast becoming the standard language used in commercial DBMS (Date 1981, pp. 145–58; Date 1987; Fleming and van Halle 1989). SQL is particularly simple. A handful of commands allows users to "query" (interrogate) a database in very general ways and without any need for programming.

SQL provides two basic commands for data retrieval operations: *select* and *from*. The *select* statement tells the DBMS which columns (fields) to retrieve; the *from* statement tells the DBMS the name of the table where the column is to be found. The set of commands necessary to perform an operation of data retrieval is called a query. The syntax of the two commands is

```
select column name(s)
  from table name
```

Thus, the query

```
select actor_name
  from actor_table
```

provides a list of all the nineteen actor names (subjects and objects) available in the table of actors: <actor_name> police officer, motorcyclist, ? (crowds), police officers, police officers, man, police officers, man, crowds, ? (crowds), crowds, crowds, ? (crowds). . . . To eliminate duplicate entries, we can qualify the query with the *distinct* statement. Thus the query

```
select distinct actor_name
  from actor_table
```

results in the following list of eight unique actors involved in the Miami race disturbances:

<actor_name> police officer, motorcyclist, ? (crowds), police officers, man, crowds, ? (ambulance), people

Names can also be sorted (in ascending or descending order) or grouped in specific ways. More generally, queries can be qualified—that is, subjected to search conditions—via the *where* statement. Thus

```
select actor_name
  from actor_table
    where actor_name = 'crowds'
```

yields the following list: <actor_name> crowds, crowds, crowds.

The *count* function within a select clause is particularly useful for our purpose. The *count* function provides a tally of the number of occurrences of elements in a given column. For instance, the query

```
select number = count(actor_name)
  from actor_table
```

tallies the elements contained in the column actor_name and puts the result in a column labeled number: <number> 19.

A conditional query with the *count* statement could yield the number of strikes, of rallies, etc. Thus, with a simple query, we can obtain numbers from text input.

Single table operations, such as the ones described so far, can be quite useful. But the power of SQL stems from its ability to operate on several tables at a time. As we have seen, in a database environment each file or table contains information on one topic only. For instance, the action table stores information on actions, the space table on the location of actions. To find out where each action occurs, we need to join the two tables via a common field (or column or attribute)—for instance, the action_id between the action and space tables. On the basis of the common action_id field, the following query provides a list of where each action in the Miami disturbances occurred:

```
select action_table.action_name, space_table.city,
space_table.neighborhood
 from action_table, space_table
     where action_table.action_id = space_table.action_id
```

```
action_name               city    neighborhood

engage in disturbances    Miami   Overtown
engage in disturbances    Miami   Liberty City section
respond to gunfight       Miami   street, racially intense
                                  neighborhood
spread disturbances       Miami   Overtown
spread disturbances       Miami   Liberty City section
```

Relations between two tables can also be established via a third table. For instance, in our database design, actors (subjects or objects) and actions are related to one another via the triplet table. The triplet table contains the identifiers for the subject, the action, and the object. If we wanted to find out how the "crowds" behaved in the Miami disturbances, we would need to join three tables: the actor table to find out in which triplet (triplet_id) the actor "crowds" is present as a subject; we would then use the selected triplet_id(s) to match the column triplet_id in the triplet table; we would extract the column action_id(s) from those rows; and, finally, we would match the action_id from the triplet table with the action_id in the action table to obtain the action names. The following query would compactly express all of this:

```
select action_table.action_name
 from triplet_table, actor_table, action_table
     where actor_name.actor_table = 'crowds' and
         actor_id.actor_table = subject_id.triplet_table and
         action_id.triplet_table = action_id.action_table;
```

Indeed, the real power of SQL is that of allowing users to perform highly complex queries by joining several tables at a time. Thus, for instance, the instrument_table of the Miami disturbances contains two columns (action_id and instrument_name) and one row with the following entries: 10, "stone." If we were interested in finding out which actors use the instrument "stone," we could achieve this goal via the following query, involving four different tables (triplet, actor, action, and instrument):

```
select actor_table.actor_name
  from triplet_table, actor_table, action_table, instrument_table
    where instrument_name.instrument_table = 'stone' and
      action_id.instrument_table = action_id.triplet_table and
      actor_id.triplet_table = actor_id.actor_table;
```
[11]

The query would give the following results: <actor_name> ? (crowds).

6. EXTRACTING NUMBERS FROM A TEXT DATABASE ON INDUSTRIAL CONFLICT

To appreciate the power of the substantive results that we can obtain from text data organized in relational form, let me set aside the simple example of the previous sections and use, instead, a real database. The database that I will use contains information on industrial conflict in Italy taken from newspaper articles from January 1 to June 30 1986 (for substantive information on industrial conflict in Italy, see Franzosi 1994). The semantic grammar upon which the database was built consists of Actors (subject or object) and Actions, five Actor Modifiers (number of actors, type of actor, space, firm, union), seven Action Modifiers (number of actions, type of actions, space, time, demands, outcomes, instruments), and three hierarchical levels (semantic triplets, events, disputes). Each entry in the grammar has a corresponding table in the database (e.g., actions are stored in the action_table). In order to analyze hierarchical levels within a relational data model, I linearized hierarchical levels (dispute, event, and triplet)—that is, I assigned a unique identified (_id) to each unique entry in the dispute_table, even_table, and trip-

[11]In the query, we select from the instrument_table the row(s) with "stone" as instrument_name, we take the action_id from that row of the table in order to match it with the action_id in the triplet_table. In that row we find the identifier for the actor, and we use that identifier to match it with the actor_id in the actor_table to extract the actor_name.

let_table, and carried these identifiers within each table. Thus, contrary to the database design for the 1991 Miami disturbances, each table contains all the identifiers of all the higher level aggregates in the grammar (e.g., the actor_table contains a field for the triplet_id, event_id, and dispute_id; similarly, the demand_table, where <demand> is an action modifier, contains a field for the action_id, triplet_id, event_id, and dispute_id).

Setting up each hierarchy as a separate table and providing one overlapping field only between contiguous tables in the tree structure reduces disk overheads but results in quite inefficient queries. If we were interested in the demands that specific unions make in various labor disputes, several joins would be required. Upward, one would have to climb the tree, table by table, all the way to the dispute_table. This involves joining the union_table and the actor_table using the actor_id (to obtain the triplet_id); the triplet_table and the event_table on the basis of the triplet_id to obtain the event_id; etc. Downward, one would have to join the action_table to the demand_table using the triplet_id as a relation between the two tables. Several joins could be saved if each table carried the identifiers of each of the upper-level nodes in the tree. Although this approach requires more disk storage, it makes queries much faster.

There are several substantive questions that researchers may address using this database. For instance, they may query the database to find out the public's reactions to strikes, particularly service sector strikes. They might ask What do users do when they go to the bank or to a department store and find them closed due to an unannounced strike? What do they do if they are ready to go home from work and subways or buses are not running due to a wildcat strike of municipal workers? The following SQL query would create a list of all the actions performed by the "public:"

```
selection action_table.action_name
  from action_table, actor_table
    where actor_table.actor_name = 'public'[12] and
```

[12]If other synonyms such as "citizens," "people," "customers," or "users" had been used for the word "public," then we could list all these synonyms in the query. This clearly illustrates how data recoding should be done at the time of data analysis rather than of data collection.

```
actor_table.triplet_id = action_table.triplet_id and
actor_table.event_id = action_table.event_id and
actor_table.dispute_id = action_table.dispute_id;[13]
```

Using the *count* function, from the list of actions that the query produces, we could obtain a frequency distribution of the public's actions with the following results:

frequency	action_name
1	road block
1	building occupation
1	rally with speech
1	boycott (at Standa Supermarket in Rome)
2	protest against strikers
3	assembly
3	sign petitions of solidarity with other workers
4	meeting
4	strike
7	rally
11	demonstration

The results show that the public plays quite an active role in industrial conflict in Italy, sometimes marching along with the workers, sometimes expressing solidarity to strikers in a variety of ways (e.g., signing petitions or boycotting a store), and yet sometimes reacting angrily toward strikers.

Again, a researcher may be interested in the type of demands and grievances that workers voice. The following query would address this concern:

```
select frequency = count(distinct dispute_id),[14] demand_name
  from demand_table
    group by demand_name;
```

[13]The query restricts the choice of actions to those actions that belong to the same triplet, event, and dispute of the actor "public."

[14]The statement (distinct dispute_id) implies that the unit of counting is the dispute (dispute_id) and that each dispute should be counted only once (distinct), regardless of the number of times the same expression for demand_name appears within the dispute.

```
frequency  demand_name
  . . .      . . .
   71        work organization
   79        plant closing
   85        more hiring
   89        defense of employment
   99        firm restructuring
  118        unemployment benefits
  118        wage & economic demands
  133        resolution of the dispute
  134        layoffs
  146        management's acknowledgment of workers' demands
  146        contract renewal
  155        firm crisis
```

The frequency distribution of demands seems to point to a labor movement on the defensive in the mid-1980s. Aside from demands centering on issues of contract renewal, the demands most often voiced are firm crisis, layoffs, unemployment benefits, plant closings, and plant restructuring. Only two demands seem to be at odds with a defensive stance: "more hiring" and "wage & economic demands." These results are in line with similar results based on official strike statistics: one of the clearest findings of quantitative strike research is that industrial conflict is negatively related to the level of unemployment (Franzosi 1994). Indeed, in 1986, the labor market in Italy was quite unfavorable. The process of industrial restructuring of the late 1970s and early 1980s was slowing down, but unemployment was soaring.

Service economy and postindustrial society theories argue that service conflict is insensitive to the fluctuations in the level of economic activity. Is there any evidence in our database that industrial and service sector workers responded differently to the crisis? For instance, did the number of industrial disputes decrease and that of service disputes increase over previous years? The following two queries produce a frequency distribution of labor disputes in the two sectors:

```
select frequency = count(distinct dispute_id), sector_name
  from sector_table
    where sector_name = 'industry';[15]
```

[15]We are tallying each dispute only once, regardless of how many times a particular sector name appears in the account of the dispute.

```
frequency   sector_name

577         industry
```

```
select frequency = count(distinct dispute_id), sector_name
  from sector_table
  where sector_name = 'service';
```

```
frequency   sector_name

329         service
```

577 disputes in industry in the first semester of 1986 are a far cry from the several thousand disputes typical of a decade earlier. The high ratio between service and industrial sector disputes is also new. Service disputes constitute 36 percent of the total number of disputes, up from just a few percentage points of earlier periods.[16]

If indeed industrial and service workers behave differently with respect to the state of the labor market, these differences should be reflected not only in the frequency of their actions but also in the nature of their grievances (demands). Industrial workers should voice defensive demands (against plant closings, against layoffs, against pay cuts, etc.), while service workers should not. To test this hypothesis, we need to obtain frequency distributions of workers' demands by economic sector (industry/service). In practice, for each sector, we need to perform two queries: the first, to extract all workers' demands from the database;[17] the second, to obtain a frequency

[16]There is evidence, however, that news coverage of labor disputes is highly biased toward service disputes (Franzosi 1987). My data may simply reflect the bias of the data source. Yet, official strike statistics report 741 strikes for industry and 724 for services in 1986. If anything, my newspaper source (*L'Unità*) seems to underreport service disputes when compared to official strike statistics. But the decrease in strike frequency in industry and its increase in services at a time of economic crisis confirm the stronger relationship between the business cycle and industrial sector conflict.

[17]The exact SQL statement is

```
select demand_table.demand_name, demand_table.dispute_id
  from demand_table, sector_table
    where sector_table.sector_name = 'industry' and
      sector_table.triplet_id = demand_table.triplet_id and
      sector_table.event_id = demand_table.event_id and
      sector_table.dispute_id = demand_table.dispute_id;
```

distribution of the demands extracted.[18] The result of these combined queries for the industrial sector is

frequency	demand name
. . .	. . .
49	resolution of the dispute
65	plant restructuring
65	management's acknowledgment of workers' demands
73	job security
100	against plant closing
112	unemployment benefits
140	firm crisis
113	against layoffs

The same set of queries for the service sector gives the following results:

frequency	demand_name
. . .	. . .
62	wage & economic demands
64	management's acknowledgment of workers' demands
69	resolution of the dispute
71	more hiring
79	contract renewal
93	quality of services provided

The results show that Italian workers, whether in the industrial or in the service sectors, seem to have a hard time in getting the counterpart to listen to their demands (see the high frequency of "management's acknowledgment of workers' demands" and "resolution of the dispute"). This is altogether typical of industrial relations in unfavorable labor market conditions. The results, however, also point to striking differences in workers' behavior in the industry and service sectors. Service workers show a concern with the efficiency of the service provided and demand improved quality of service for the users. On the one hand, this may reflect a real level of inefficiency of

where, again, the query is restricted to demands within the same triplet, event, and dispute.

[18]We could accomplish this using the *count* function.

Italian public services. In Italy, many services (e.g., transportation, communication, health, education) are run by the government, more with the "political" allotment of jobs in mind than economic efficiency. On the other hand, these demands may be part and parcel of an overall strategy by service sector workers and their unions to win over the support of a public otherwise often hard-hit by service workers' actions (e.g., during a train or bus strike). But, most importantly, service workers voice offensive demands (e.g., "more hiring," "wage & economic demands"). Industrial workers, on the other hand, are on the defensive. The most frequent reasons given for striking in industry are "job security," "against plant closing," "against layoffs," and "firm crisis."

The workers' market situations vary not only across economic sectors but also within them. The majority of industrial workers belong to the private sector; service workers are concentrated predominantly in the state sector. Whether at the local or at the central level, the state is less likely than private employers to resort to layoffs. If the state follows Keynesian economic policies in order to foster demand, during economic downturns it may actually expand rather than reduce the size of public employment. The locus of employment (private or public), or type of ownership, may thus be a more crucial axis of differentiation in workers' behavior than the economic sector. In order to test this hypothesis, we could relate the type of demand (offensive/defensive) to the type of ownership (private/public). This would require dichotomizing both "type of demand" and "type of ownership" variables, with values of zero or one (zero = defensive or private; one = offensive or public).[19] Once we

[19]To transform multinominal text variables ("against layoffs," "contract renewal," etc.) to binomial numeric variables (0,1), we would need, for instance, to create a new demand_table with a new attribute (demand_dummy) via a *create table* statement, to copy into the new table the values of the old table, and then to assign numeric values to the new column via *update* statements of the kind

```
update new_demand_table
  set demand_dummy = 0
    where demand_name in ('plant closing', 'layoffs', . . .)

update new_demand_table
  set demand_dummy = 1
    where demand_name in ('more hiring'; 'wage demands', . . .)
```

TABLE 1
Frequency Distribution of Disputes by Demands and Ownership

Ownership	Demands		
	Defensive	Offensive	Total
Private	546	125	671
Public	29	206	235
			906

have constructed these variables, we can organize the empirical evidence in a two-by-two table of demands (offensive/defensive) versus ownership (private/public). According to the results of Table 1, while 81 percent of demands are defensive in the private sector, only 12 percent are in the public sector.

We could summarize the univariate and bivariate arguments presented thus far in a multivariate model, where the type of demands voiced is expressed as a function of both economic sector and type of ownership:

Model (1) type of demand = f(economic sector, type of ownership)

The level of aggregation of the model is the labor dispute. The results of a logit regression using these data are shown in Table 2.

The multivariate results strongly support the hypothesis that the location of conflict (industry versus service, private versus public) is an important predictor of the nature of the demands (defensive versus offensive) voiced by Italian workers in labor disputes. The probability of offensive demands is much higher (.88) in public services than in other settings.

The results of the maximum likelihood estimates of model (1) provide important substantive insights. But, even more important, at least in the context of the methodological problem dealt with in this paper, is the fact that we were able to perform multivariate statistical analyses starting from words as input. In other words, we were able to go "from words to numbers." Furthermore, we could not have obtained these results without organizing the data in complex, rela-

To obtain the type of ownership variable, we would apply a similar procedure.

TABLE 2
Logit Regression Estimates

	Estimate	Standard Error
Intercept	−2.089	0.133
Economic Sector	2.675	0.247
Type of Ownership	1.417	0.286

Data

Intercept	Economic Sector	Type of Ownership	Proportion Offensive
1	Service (1)	Public (1)	0.88
1	Service (1)	Private (0)	0.21
1	Industry (0)	Public (1)	0.34
1	Industry (0)	Private (0)	0.11

tional structures. Counting words is certainly possible even with traditional coding schemes, with their rectangular data organizations. From these schemes it is also possible to extract basic crosstabulations. By and large, however, traditional content analysis data are highly aggregated. They are also organized into separate and unconnected tables. Analyses at lower levels of aggregation and general crosstabulations between any data item are thus impossible.[20] The methodology used here, based on relational structures, allows a researcher to provide an endless number of crosstabulations, each at highly disaggregated levels.

7. CONCLUSIONS

Semantic grammars, based on the linguistic structure Subject/Action/Object (SAO) and their modifiers, provide powerful coding schemes for the collection of narrative data. These schemes offer several advantages over traditional content analysis schemes, among them: (1) a grammar preserves both the connection between the elements of the language (e.g., actions are related to actors, modifi-

[20]This is not only true for official statistics, whether on industrial conflict, race disturbances, or other matters, that are basically in numeric form. It is even more true for textual data (e.g., medical reports, newspaper articles) where the basic information itself is already in relational form.

ers are related to both actors and actions) and much of the original text input (i.e., coded output is predominantly made up of words); and (2) a grammar produces larger and yet more reliable volumes of information.

We may wonder whether the advantages of a linguistic approach to content analysis based on text grammars may actually turn to disadvantages, in terms of the complexities of data organization (how do we code and store all this highly interrelated information?) and of data analysis (how do we analyze words statistically?).

In this paper I have argued that set theory has a solution for both problems. I showed that a semantic grammar for the collection of text data can be reformulated as a set theoretical problem. The advantages of using a set theoretical framework for the grammar are twofold. First, the underlying mathematical basis for Relational DataBase Management Systems (DBMS) is set theory; it is these systems that make possible a computer application of the grammar and, therefore, the collection of large volumes of text data and the organization of these data into complex structures. Second, the set theoretical concept of cardinal numbers provides the basis for tallying words—that is, for the statistical analysis of narrative data.

Set theory, however, is not merely a fancy cover for very mundane practices (namely, counting words). Because set theory is behind the coding schemes used to collect text data (text grammar), and is behind the models used to organize them (relational structures) and analyze them (cardinal numbers), set theory provides a single unifying thread and a general framework for the various phases of a linguistic approach to content analysis. It is precisely because we can rewrite a grammar in set theoretical and relational terms that we can use such mundane tools as Relational DBMSs to collect, organize, and query even complex text data, making a linguistic approach to content analysis well within the reach of any PC user.

APPENDIX: SOFTWARE CONSIDERATIONS

Content analysis software has become increasingly sophisticated, moving away from simple word frequency counts to listings and frequency counts of key words in context (KWIC), co-occurences, and concordances (e.g., see SPAD-T, TACT, TEXTPACK; on computer-assisted content analysis, see Weber 1990). Yet the context for word counts of even the most powerful currently available automated content analysis software is quite limited and rigid (namely, context is hierarchically structured into either sentences or paragraphs). Furthermore, many of these software packages require extensive manipulation of text input (e.g., SPAD-T, TEXTPACK, TACT).

Similarly, artificial intelligence (AI) has come a long way in its work on computer understanding of natural languages. Computational linguistics has developed several algorithms that can successfully process text. Yet currently available AI language parsers perform well only within particular sublanguages (e.g., technical or scientific reports, newspaper articles on industrial conflict), where vocabulary is limited and syntactic structures simple (in particular, of the SAO type), such as medical records (e.g., see Sager 1987). In any case, many AI language parsers require extensive manipulation of text input, in a manner not dissimilar from automated content analysis software. For instance, in RELATUS, an AI program developed at MIT, the user has to rewrite input text into a format acceptable to RELATUS, simplifying the syntactic structure, excluding all figures of speech such as metaphors (Alker et al., 1991, pp. 102, 121). Furthermore, the user has to supply an extensivé semantic set of background beliefs and comments.

A semantic grammar approach to coding can be easily implemented in a relational database environment. Commercial DBMSs for PCs are now widely available (e.g., Oracle and Ingres and even simpler packages such as DB4, Paradox, and Access). Some statistical packages (e.g., SAS, SPSS) are starting to offer an integrated environment for the analysis of complex data. In these environments it is possible to submit jobs that combine data analysis statements and SQL statements for the manipulation of data organized in relational structures. This greatly simplifies the process of data analysis. Nonetheless, both data entry and data retrieval can be quite slow (Solenberger et al. 1989, p. 201). In large-scale data collection projects, this may add considerably to research costs (Franzosi 1990*b*).

Custom designed and special purpose software can greatly reduce data entry time and cost. For instance, in PC-ACE (Program for Computer-Assisted Coding of Events), a data entry program strictly based on a semantic grammar that I developed for a large-scale project on industrial conflict in Italy (Franzosi 1990*b*), coding time varies between 3 to 10 minutes per newspaper articles, depending upon the coding technique chosen, the length of the article, and the coder's experience. However, researchers should be aware of the high costs of large software development (Doyle 1989, p. 187; Winsborough 1989, pp. 245–47).

REFERENCES

Alker, R. Hayward, Gavan Duffy, Roger Hurwitz, and John Mallery. 1991. "Text Modeling for International Politics: A Tourist's Guide to RELATUS." In *Artificial Intelligence and International Politics,* edited by Valerie Hudson. Boulder: Westview Press.

Benzecri, J. P. et al. 1980. *Linguistic et lexicologie: Pratique de l'analyse des données. Tome 3.* Paris: Dunod.

Date, J. 1981. *An Introduction to Database Systems.* Reading, Mass.: Addison-Wesley.

———. 1986. *Relational Database Selected Writings.* Reading, Mass.: Addison-Wesley.

———. 1987. *A Guide to the SQL Standard.* Reading, Mass.: Addison-Wesley.

David, Martin. 1989. "Managing Panel Data for Scientific Analysis: The Role of Relational Data Base Management Systems." In *Panel Surveys,* edited by Daniel Kasprzyk, Greg Duncan, Graham Kalton, and M. P. Singh, 226–41. New York: Wiley.

Doyle, Pat. 1989. "Data Base Strategies for Panel Surveys." In *Panel Surveys,* Daniel Kasprzyk, Greg Duncan, Graham Kalton, and M. P. Singh, 163–89. New York: Wiley.

Fleming, Candace C., and Barbara van Halle. 1989. *Handbook of Relational Database Design.* Reading, Mass.: Addison-Wesley.

Franzosi, Roberto. 1987. "The Press as a Source of Socio-Historical Data: Issues in the Methodology of Data Collection from Newspapers," *Historical Methods* 20 (no. 1):5–16.

———. 1989. "From Words to Numbers: A Generalized and Linguistics-Based Coding Procedure for Collecting Event-Data from Newspapers," In *Sociological Methodology, Vol. 19,* edited by C. Clogg, 263–98. San Francisco: Jossey-Bass.

———. 1990*a*. "Strategies for the Prevention, Detection, and Correction of Measurement Error in Data Collected from Textual Sources," *Sociological Methods and Research* 18, (no. 4):442–72.

———. 1990*b*. "Computer-Assisted Coding of Textual Data Using Semantic Text Grammars," *Sociological Methods and Research* 19 (no. 2):225–57.

———. 1994. *The Puzzle of Strikes: Class and State Strategies in Postwar Italy.* Cambridge: Cambridge University Press.

Halmos, Paul R. 1960. *Naive Set Theory.* Princeton, N.J.: D. Van Nostrand.

Lebart, L., and A. Morineau. 1984. *SPAD, Tome III, Analyse des données textuelles.* Paris: Cesia.

Sager, Naomi. 1981. *Natural Language Information Processing, A Computer Grammar of English and Its Applications.* Reading, Mass.: Addison-Wesley.

———. 1987. "Computer Processing of Narrative Information." In *Medical Language Processing: Computer Management of Narrative Data,* edited by Naomi Sager, Carol Friedman, and Margaret S. Lyman, 3–22. Reading, Mass.: Addison-Wesley.

Sager, Naomi, Paul Mattick Jr., Carol Friedman, and Emile C. Chi. 1987. "Information Structures in Survey Instruments," *American Statistical Association: Proceedings of the Section on Survey Research Methods,* 267–70.

Shapiro, Gilbert, and John Markoff. Forthcoming. *Revolutionary Demands: A Content Analysis of the Cahiers de Doléances of 1789.* Stanford: Stanford University Press.

Solenberger, Peter, Marita Servais, and Greg J. Duncan. 1989. "Data Base Management Approaches to Household Panel Studies," In *Panel Surveys,* edited by Daniel Kasprzyk, Greg Duncan, Graham Kalton, and M. P. Singh, 190–225. New York: Wiley.

Ullman, Jeffrey D. 1980. *Principles of Database Systems.* Rockville Md.: Computer Science Press.

Weber, Robert P. 1990. *Basic Content Analysis.* Newbury Park, Calif.: Sage.

Winsborough, Halliman H. 1989. "Data Base Management: Discussion." In *Panel Surveys,* edited by Daniel Kasprzyk, Greg Duncan, Graham Kalton, and M. P. Singh, 242–48. New York: Wiley.