Structural Methods for Lexical/Semantic Patterns

Scott A. Waterman

Brandeis University
Computer Science Dept.
Waltham, MA 02254-9110
email: waterman@cs.brandeis.edu

ABSTRACT

This paper represents initial work on corpus methods for acquiring lexical/semantic pattern lexicons for text understanding. Recently, implementors of information extraction (IE) systems have moved away from using conventional syntactic parsing methods, instead adopting a variety of *pattern based* approaches for complex IE tasks. While there has been much work towards automated acquisition of lexicons for conventional syntactic processing, little progress has been made towards those for pattern systems, due primarily, in the author's opinion, to a lack of a linguistic framework in which to view their use. In combining a functional view of both denotational semantics and syntactic structure, this paper provides a basis for examining the structural constraints between the two. This functional viewpoint is the starting point for methods to investigate the characteristics of the interaction between text and denotation, from the perspective of pattern-based systems. An approach for determining and exploiting these structural constraints is outlined in terms of building hierarchical lexical structures for text understanding. Experiment results for such a method are given, demonstrating the functionality of the approach.

1. Introduction.

Recently, implementors of information extraction (IE) systems have moved away from using conventional syntactic parsing methods, instead adopting a variety of *pattern based* approaches for complex IE tasks (such as the MUC contests and the ARPA-sponsored TIPSTER research). These pattern-based systems use short and fairly specific lexical patterns to specify the relation between strings in the source text and particular entries in a problem-dependent knowledge representation. This one-step process substitutes for a conventional two-level process of a full syntactic parse followed by semantic interpretation. With considerably less time and development effort (notably demonstrated by [11, 8]), these systems achieve performance comparable to more standard systems that rely heavily on full syntactic analysis ([9, 5]). However, because these pattern-based systems are still viewed as linguistically ungrounded and somewhat *ad hoc*, formal work in the application and acquisition of lexical patterns has lagged system development. In most current systems, patterns are produced through tedious hand analysis of text ([11, 4, 8]), while system coverage is gained either through extensive linguistic knowledge on the part of the researcher (in judging appropriate pattern generalizations), or by generating and testing massive numbers of patterns.

One exception to hand analysis is Lehnert's work in [12], in which machine learning techniques are used to infer possible patterns for extraction. While this *AutoSlog* technique has dramatically reduced system development time, the inference techniques use only sparse linguistic information, provide no means of generalizing patterns across domains, and still require that the rules be checked by the researcher for applicability.

1.1. A theoretical framework

By placing pattern-based approaches in a lexical semantic framework, such as Pustejovsky's Generative Lexicon theory ([16]), my aim is to provide a basis for pattern-based understanding systems which can be used to relate pattern-based approaches to more conventional linguistic approaches. Within such a framework, methods for pattern acquisition can be studied and developed and the effectiveness of patterns can be assessed.

My main contention is that this framework can be developed by viewing the lexical patterns as structural mappings from text to denotation in a compositional lexical semantics, merging the distinction between syntactic and semantic analysis, and obviating the need for separate syntactic and semantic processing systems. This interpretation follows directly from an appeal to functional semantic principles. In the framework I present, patterns indexed to individual words relate semantic interpretations to lexical constraints, in a manner dictated by context. Patterns for multiple words in context can be combined to provide a consistent interpretation for large constructions — a mechanism that could be viewed as a lexically distributed semantic grammar.

A combined approach to pattern acquisition is outlined here, with two orthogonal methods whose combination leads to the construction of organized sets of lexical/semantic patterns which relate strings in the source text to denotations in a predicate structured knowledge representation. A key feature of the mechanism is that these resulting patterns are organized hierarchically in terms of the specificity of both syntactic and semantic constraints.

The methods presented are based on the *structural* properties of the text and denotation, attacking the problem of pattern acquisition from the separate directions of a purely syntactic approach and a purely semantic approach. The natural merger of the two methods results in an automatic machine learning technique for the production of lexical/semantic patterns, relating use in context to denotation.

Related works describing corpus techniques for deducing lexical structure ([18, 19]) and semantically marked selectional frames ([6]) suggest that lexical/semantic patterns can be induced from corpora, but do not directly apply to the generation of these distributed patterns.

2. Information Extraction

The recent Message Understanding Conferences (MUCs) and ARPA Tipster project have posed a complex and fairly specific problem in information extraction (IE). The problem given is that of creating semantic *templates* or *frames* to correspond to newswire and newspaper articles. The expressiveness of the templates is restricted and somewhat skeletal, capturing the bare facts of the text, and not its complete meaning. Hobbs ([8]) has argued effectively that the problem is not one of full text *understanding*, but specifically one of *information extraction* — many types of information, such as speaker attitude, intensional constructs, facts not relevant to the chosen domain, etc., are not required; only a representation-specific set of domain information is the target for extraction.

These types of systems provide a useful groundwork for the study of text interpretation systems because of the relative lack of difficulty in representing and manipulating the resulting knowledge structures. Although denotational structures for the type of factual information required in IE can be quite complex, they are still far more tractable than representations of speaker attitude, opaque contexts, or intensional constructions.

For example, in the ongoing TIPSTER project, information in only two specific domains is to be extracted – one domain is joint ventures and business ownership, the other the microelectronics industry. The domains are further restricted by the particular hierarchy of predicate types used in the knowledge representation. Each domain has a set of *templates* (a particular implementation of frames) which rigidly define what types of facts and relations from the text are representable.

2.1. Mapping — IE : text$\mapsto$KR

These information extraction tasks, as a subset of text understanding tasks, can be viewed as *mapping problems*, in which the problem is to find the proper representation, in terms of templates, for the source text. The problem is one of mapping from the strings of the source text to a problem-dependent knowledge representation scheme.

The template knowledge representation used in the TIPSTER/MUC tasks is based on a frame-like system commonly known as the *entity-relation*, or ER, model.

The ER model codes information as multi-place *relations*. Typically, each type of relation has a fixed number of arguments, each of which is an *entity* in the model. Entities can either be *atomic* — in the case of TIPSTER, atoms can be strings from text or items from a predetermined hierarchy of types — or they can be *composite*, referring to other relational structures.

Objects referenced in text often participate in more than one relationship. For example, the direct object of a sentence will often be the subject of a subordinate clause, either explicitly, or by pronominal reference. In a strict ER model, this direct object would have to be represented twice, once for each clause. By a slight extension, atoms in the ER model can be generalized to objects which can take multiple references. Thus, no real atoms appear in relations, but only references to atoms, or to other relations. This model is often termed an *object-oriented* model, but because of the overloading of that name in so many fields, I prefer to call these models *reference-relation* models (RR). The important extention from the ER model is that relations themselves may be treated as objects of reference by other relations.

3. Lexical Semantics

The structure of the denotational representation is important not only for its expressiveness, but also in its relationship to the structure of the language it is to be derived from. In part, the structure of the language is determined by the semantic constraints of relations that are conveyed by its use. If the model is accurate enough, these constraints will be reflected in the representation.

Many, if not most, semantic theories used in computational linguistics today assume some degree of *functionality* in language — words act as *operators*, or take *arguments*, or act as *logical quantifiers* over the things denoted in their context. The corresponding grammatical theories (e.g. CFG, LFG, HPSG, GB) assume a parallel functional structure, incorporating notions of *combinational categories*, *argument structure*, or *selectional frames* into the lexical representation. These structures use individual words or constituent phrases as functional objects, projecting *expectations* or *constraints* to subcategorize for *arguments* to be incorporated into the function.

3.1. Structurally specified semantic relations

The functional semantics of the operators, then, specify the nature and lexical appearance of the arguments. The appearance of a particular head will generate expectations for the number and kind of arguments to be found, and dictate the semantic relation to be applied to them — because we have seen the operator, we can expect to find its operands in the vicinity. Further, if these operands do not have distinct types, we will need some other mechanism, such as position or order, to be able to distinguish them. In this way, the need for syntactic structure is driven by typing ambiguities in the semantics.

There is an immediate parallel between the semantic specification of function/argument structure and the specification of the reference-relation representations: the function is analogous to the predicate relation, while the arguments are the referenced components of the relation. In computational linguistic models, this sort of functional semantics has proved very useful in providing a mechanism for deriving frame-like denotations when processing language (predicate logic and unification frames, two of the more popular denotation schemes, can both be transformed to general RR models). In fact, it is often the case that the relations of the RR model are the same as the semantic relations specified by the language. (Whether this is because of a desire for representational efficiency or for other reasons I will leave unexplored.)

Semantically specified structural interpretation: We can rephrase the relation between a functional head and its arguments in the following way: since the head requires a particular semantic relation to its arguments, an argument with an otherwise ambiguous interpretation must be treated as being of the type required by the head. Because we know the interpretation of the operator, we can constrain the various arguments to have a corresponding and consistent interpretation.

This type of argument disambiguation is exhibited in the phenomenon of *type coercion* ([16]).

3.2. The syntax–semantics boundary

In terms of the function-argument structure or reference-relation representations, words or categories with similar type amibiguities and similar argument number are described as being syntactically similar, while differing in interpretation. On the other side, categories with similar functional or relational type are said to have similar semantics, even though the number and typical realization of arguments might differ considerably.

As the specificity of the relational constraints varies, the distinction between the two can also vary. Some highly cased languages (e.g. Japanese and Latin) have loose syntactic constraints; the case marking develops constraints for the consistent semantic incorporation of the various arguments within the functional scope of the heads. Other languages, such as English, have a much more definite word order, where the configuration of arguments and heads constrains their semantic relationships. Some constructions, such as idiomatic expressions, have both completely a fixed syntax *and* semantics. Poetic use has both a freedom of word order and a loose interpretation. Each form of linguistic construction, however, has a consistency of interpretation derived from its components.

By using a mechanism of language interpretation that explicitly examines the *degree of specificity* in argument position and in argument type, and especially their interaction with one another in use, one should be better able to achieve the goals of interpretation; that is, to relate the text to a particular denotation.

4. The Generative Lexicon

Theoretical approaches to lexical semantics have begun to incorporate this merging of syntactic and semantic description. The incorporation of argument structure or selectional frames is a large step in this direction. While the notion of argument structure is usually reserved for verbs, some theories, such as Pustejovsky's generative lexicon (GL), extend the idea to include *all* lexical categories ([16, 17]). For the purposes of this discussion, we can consider the GL lexicon to carry two sorts of selectional information with every term:

- Qualia structure, which provide semantic type constraints on arguments. These constraints are used both in deriving expectations for the syntactic form of arguments, and in *coercing* ambiguous or polysemous arguments into the required types ([16]).
- Cospecifications, which constrain the syntactic realizations and ordering of arguments in relation to the lexical entry and to each other. These constraints

are specified much like regular expressions, and can provide varying degrees of 'fit' to the syntax.

In addition to these selectional constraints, each term has a mapping from the arguments to a predicate logic denotation, detailing the relationship in which the arguments participate.

These three together embody what Pustejovsky calls a *lexical-conceptual paradigm* (LCP), a representation of the expression of a particular concept, and the paradigmatic usage in context of the lexical entry to express that concept ([19]).

It is easy to see how a theoretical approach such as GL can be operationalized: A local grammar, corresponding to the cospecifications, and indexed off the lexical entry, could be used in conjunction with a type matching system which imposes the semantic constraints of the qualia structure. The resulting mechanism, when matched against text, could place the matching arguments appropriately in the predicate denotation to return an interpretation of the text.

This system, which by conjoining argument type and positional information avoids making a distinction between separate syntactic and semantic analysis, would be a *pattern system.*

This system has been implemented, in part, in the DIDEROT information extraction system ([4]).

5. Patterns

Pattern-based extraction systems combine syntactic and semantic processing through the use of *patterns*. Patterns consist of lexically specified syntactic templates that are matched to text, in much the same way as regular expressions, that are applied along with type constraints on substrings of the match. These patterns are lexically indexed local grammar fragments, annotated with semantic relations between the various arguments and the knowledge representation. In the most general system, the units of matching could range from single lexical items to phrasal components or variables with arbitrary type constraints. The variables in the pattern can be mapped directly into the knowledge representation, or, through type constraints, used as abstract specifications on the syntax. Pattern-based systems operate by combining numerous local parses, without relying on a full syntactic analysis.

5.1. DIDEROT, a pattern example

For example, in the DIDEROT project ([4]), a pattern is represented as a GL structure (GLS) which gives the syntactic context along with mappings from text variables to an predicate logic knowledge representation. A typical set of patterns used to extract joint-venture events, indexed here from the word 'establish', is given in figure 1.

```
gls(establish,
  syn(...),
  args([arg1(A1,
      syn([type(np)]),
      qualia([formal([code_2,organization])])),
    arg2(A2,
      syn([type(np)]),
      qualia([formal([code_2,organization])])),
    arg3(A3,
      syn([type(np)]),
      qualia([formal([code_2,joint_organ])]))]),
  qualia([formal(tie_up_lcp)]),
  cospec([
    [A1,*,self,*,A2,*,with,A3],
    [A1,and,A3,*,self,*,A2],
    [A1,together,with,A3,*,self,*,A2],
    [A2,is,to,be,self,*,with, A3],
    [A1,*,signed,*,agreement,*,self, A2],
    [A1,*,self,*,joint,venture,A2,with,A3],
    [self,include,A2],
    [A2,was,self,with,A3]]),
 types(tie_up_verb),
 template_semantics(pt_tie_up,
    tie_up([A1,A3],A2,_,existing,_))).
```

Figure 1: A GLS for 'establish'

The GL cospecification information is contained in the **cospec** field. The index variable '**self**' is used to refer to an appearance of any of the morphological forms of 'establish'. These forms are given in the **syn(...))** field (omitted here for brevity). Literals, such as 'venture' or 'agreement' must match the text exactly. The **args** field indicates that argument variables **A1** and **A2** must be realized syntactically as **type(np)**, where **np** designates a class of strings which are heuristically noun phrases. The argument variables are further restricted to the semantic *type path* **[code_2,joint_organ)]**. The type path establishes a region in a type hierarchy which must contain the type of the argument ([20]). The last component of the cospec, '*****', is a Kleene star over all tokens — anything or nothing may appear in this position.

Because of the difficulty and expense of deriving patterns, GLSs cannot be produced for every term of importance. Rather, large segments of the lexicon are statically typed in a sublexicon less intricate than the GLS lexicon. When the GLS is applied to text, the matching of argument variables is accomplished either by calls to GLSs of the appropriate type, or by the invocation of small heuristic grammars. These small grammars combine the type information of their constituents to match the constraints of the governing GLS.

These grammars are used especially for proper name recognition. Both company names and human names are matched using small grammars based on part-of-speech tags and the sublexicon typing. Some company names are keyed from semantic indicators such as 'Corp.' and 'Inc.', while many human and place names are identified from a large fixed name lexicon.

Overall, other pattern-based systems operate in much the same manner, varying somewhat in the amount of machinery for pattern-matching, and the richness of the typing systems.

6. The current state of Pattern Acquisition

The TIPSTER and MUC projects have provided a wealth of knowledge about building pattern-based systems. The hardest and most time-consuming task involved is certainly the acquisition of patterns, which is still done primarily by tedious hand analysis. Working backwards from the *key* templates (hand generated knowledge representations of texts as interpretted by the project sponsors), one can, by careful reading of the text, usually find those segments of text which corresponds to the representation entries Although the key templates are originally created by a researcher doing a careful reading, the correspondence between text segments and the key templates has not been recorded, making the process error prone and leaving the text open for reinterpretation. The next step, that of correlating the text with the representation and deriving a pattern which captures the relation, is the most tedious and difficult part of the task. Typing constraints for each class of predicate must be remembered by the researcher performing the task, and interactions between patterns must be identified and analyzed for possible interference.

Here is a short (and most likely incomplete) review of the state-of-the-art in pattern acquisition, as it exists in the IE community:

CIRCUS (Lehnert et al. [11]) — Handwritten CN (concept node) patterns for partial template extraction. Many man-hours were spent reading text, extracting all possibly relevant contexts. Patterns were checked by running the system. A knowledge-poor method with good coverage due to large numbers of trials.

Shogun (Jacobs et al) — Handwritten AWK scripts. Derived from compiled lists of company names, locations, and other semi-regular semantic types. Also from researcher analysis of these in context. Designed to augment or replace previous methods with similar functionality.

FASTUS (Hobbs, Appelt, et al [8]) — Handwritten regular expression-like patterns for partial template extraction. Years of linguistic system building expertise improved pattern generality and helped avoid interactions between patterns.

DIDEROT (Cowie, Pustejovsky, et al [4]) — Patterns for full template extraction. Initial patterns automatically derived from structured dictionary entries [2, 25] give moderately effective high level patterns. Partly automated tuning to corpus usage. Hand analysis of contexts and addition of patterns was used to complete coverage.

CIRCUS + AutoSlog (Lehnert et al [12]) — Automated reference from template to text, using machine learning inference techniques, gives much of the coverage previously provided by hand-analysis. Patterns must still be corrected by the researcher.

The AutoSlog approach has obtained the most significant benefit from automated acquisition. In this system, a sentence containing a string which corresponds to a template entry is analyzed for part-of-speech and major phrasal boundaries. If the string entry from the template aligns with one of the phrases, a pattern is generated corresponding to the observed syntactic structure. However, since the generated AutoSlog patterns are produced from single occurrences of context patterns, they are not likely to capture patterns generalizing to varying contexts. In addition, the acquisition method is so closely tied only to specific parts of the knowledge representation (in that string entries only are matched) that extending the coverage, or generalizing the domain appears to be as difficult as porting to entirely new domains.

7. Structural Similarity Clustering

The pattern systems described here attempt to relate the use of terms in context to corresponding denotations. One of the major assumptions made here, as well as in all algorithmic computational linguistic systems, is one of *consistency* of use and meaning — that a term or phrase (or any linguistic structure) used in a particular fashion will give rise to a particular denotation. The goals of any grammar induction or lexical semantic acquisition problem are to define those particulars — to find the distinguishing features of the usage as they relate to the features of the denotation.

The approach given here chooses to focus only on the *structural* features of usage and denotation. By classifying features relevant to the text$\mapsto$denotation mapping, the aim is to provide a vocabulary and mechanism for deriving and evaluating interpretation procedures.

It has been noted already that there exist paradigmatic usages of terms to express particular concepts (the LCPs). It is not a large leap to venture also that particular concepts have paradigmatic expressions in words – idiomatic expressions, 'stock phrases' and proper names being the most obvious examples. The relationship between the two can be approached from both directions – by classifying the uses of a word in terms of their conventional expression of a concept, or by classifying the expressions of a concept in terms of the words used. These classifications create a vocabulary that can be used to compare and relate words with concepts.

This work provides a step in forming such a vocabulary by examining methods for classifying the structural properties of the words and denotations separately, and in suggesting methods by which they could be unified. Classification methods for both lexical and semantic structure are outlined here. An experimental implementation of the lexical approach is presented in the latter sections of the paper.

7.1. Lexical structure

Without considering its semantics, the use of a word can be expressed solely by its lexical environment, or context. Grammar-driven systems as well as pattern systems achieve their performance by relying on the expected structural properties of the language. We can express the consistencies and paradigms in the usage of a word in explicit terms of the similarities and common structural properties of the lexical environment in which that word appears.

A large collection of usages could be analyzed to find natural classes of context, defined purely in terms of the lexical environment, to give a vocabulary of *context types* that can be used to compare and relate differing words. The similarities of context would be determined by the structural similarities of their component strings of words. The presence and relative ordering of identical words, words belonging to the same structural similarity classes, or phrasal components, recursively defined in terms of context types, would be the environment features necessary for determining these classes.

Groups of contexts could be organized into context types based on these similarity measures, with group membership determined by similarity. The contexts could be assembled into a hierarchical structure, in which groups of high similarity combine to form higher-order clusters encompassing the structural features of their component groups.

Word classes could be defined inductively on this tree of context types by classifying words according to the sets of context types in which they have appeared. The hierarchy of context types and word classes encodes the specificity of the relation to the category. Lower levels of the hierarchy have strict context constraints, while higher levels, combining the classes beneath them, place looser constraints on context patterns. By studying the lexical context classes in relation to the semantic properties of the terms, we could illuminate those features of context which correlate with, and in theory constrain, their semantic properties.

An experimental method for performing these sorts of classification is presented in the later part of this paper, using string edit distance as a metric of similarity, and agglomerative clustering techniques to provide the classification structure.

7.2. Semantic structure

In an analogous way, the predicate denotations of text could be classified purely from their structural properties. In exactly the same manner as for context classes, relation predicates could be grouped hierarchically based on their structural features. The features one could use to derive predicate classes include predicate arity, specificity, argument types, and structure depth, as well as a semantic type hierarchy or lattice defined for specific domain.

The large databases of parallel text and denotations that would be necessary for this are not as freely available as text corpora for study. Representations would have to be generated by hand. However, the work in template filling and analysis contributed by the research community to the TIPSTER effort has shown that deriving a sufficient volume is not out of the question.

This classification of predicate structure would provide a basis for examining the constraints which predicate structure enforces on lexical realization.

7.3. Integration

The natural integration of these two lines of study would result in a vocabulary of semantic and lexical classes that would enable the correlation of the lexical structure of a text with its denotational structure, and the derivation of structural mappings between the two.

As an example of the benefits this integration might give to interpretation or IE systems, consider the following example, from the TIPSTER/MUC-5 domain:

Imagine a researcher developing the domain-dependent vocabulary for an IE system. Assume that the system has a classification of the structural properties of general text, and has also a type hierarchy for general and domain-specific representations.

The researcher has annotated a short segment of text with its interpretation in the problem domain. (See fig. 2). In the figure, the indices relate segments of text to their corresponding denotations. SMALL CAPS are used in the denotation to indicate known quantities in the domain specific type hierarchy; mixed case is used for unknown types.

$[_A[_B\text{IBM}]_B$ is jointly developing $[_C$practical X-ray tools for $[_D$the manufacture of $[_G$devices$]_G$ $[_E$based on 0.25 micron or small geometries$]_E]_D]_C$ with $[_F$Motorola$]_F]_A$.

$$\text{DEVELOPMENT}_A\left\{\begin{array}{l}\text{AGENT:}\\ \text{“IBM”}_B\\ \text{“Motorola”}_F\\ \text{PRODUCT:}\\ \text{“tools”}_C\\ \left\{\begin{array}{l}\text{TYPE:}\\ \text{X-RAY}\\ \text{USE:}\\ \text{MANUFACTURE}_D\\ \left\{\begin{array}{l}\text{PRODUCT:}\\ \text{“devices”}_G\\ \left\{\begin{array}{l}\text{FEATURE\_SIZE:}\\ 0.25\ \mu\text{M}_E\end{array}\right.\end{array}\right.\end{array}\right.\end{array}\right.$$

Figure 2: A segment of text, marked against a predicate interpretation

Now that the researcher has provided a connection between text and denotation, the system can use the classifications of context and mapping types as a vocabulary to describe the relation. For instance, it is now known that 'IBM', and also 'Motorola', can be AGENT arguments, and specifically the AGENT arguments of a DEVELOPMENT predicate. The system probably has an LCP encoding the co-agentive functionality of 'with', but now learns specifically that the DEVELOPMENT predicate allows this behavior, and that a configuration giving that interpretation is:

$[\text{A}_1 \ldots \text{PRODUCT with A}_2]$

This knowledge can augment both the LCP for 'with' and the mapping structures for DEVELOPMENT relations.

Once the system has been provided with more text-denotation pairs particular to the domain, it may find a correlation between lexical structures containing the word 'developing' and DEVELOPMENT predicate structures, and then postulate mappings between the two, building an LCP for 'developing'. Or, relying more heavily on general structural knowledge, the system could use an existing LCP for the word 'is', as represented by the syntactic pattern

$[\text{ARG}_1 \text{ is 'X' } \text{ARG}_2 \ldots]$

and the predicate structure $\begin{array}{l}\text{X-PRED}\\ \left\{\begin{array}{l}\text{ARG}_1\\ \text{ARG}_2\end{array}\right.\end{array}$

(where the word 'X' is correlated with the predicate X-PRED). This general mapping for 'is' could be used to postulate a correlation between 'developing' and DEVELOPMENT.

Only through the development of a catalog and vocabulary of structural descriptions, however, could one hope to build a system such as this.

8. Edit Distance

One method for judging the similarity between strings of lexical items (tokens) is the *edit distance* formulated by Levenshtein ([13]). This is a similarity measure based on the minimum number of token insertions, deletions, and substitutions (mutations) required to transform one string into another. A generalization of this edit distance can be made by assigning differing weights to insertions of particular tokens or classes of tokens, and by also assigning weights to token substitution pairs. Straightforward computational methods for finding the edit distance between two strings ([22, 24]) have been used on a variety of problems in biology, genetics, speech and handwriting analysis ([21]), as well as in syntactic analysis of formal languages ([14]). (For a good introduction with applications to many domains, see [21].)

To demonstrate the generalized edit distance, consider the two strings:

the path that is the path
the way that is not the way

The first string can be transformed into the second by a number of insertion, deletion, and substitution operations. Substitutions are commonly counted as two operations, since they give the same effect as a deletion-insertion combination. In this example, 'not' could be inserted; 'path' could be substituted by 'way', then the second 'path' deleted at the end, then 'way' inserted; 'that' could be deleted then reinserted, and then 'not' inserted; etc. Many different sequences lead to the same result, but there will be a minimum number of operations required for the transformation.

After a short inspection, we could expect a minimum of 5 operations in this case – two for each change from 'path' to 'way', and one for the insertion of 'not'.

This distance measure can be generalized to compensate for different similarities between types of tokens. For instance, if one decides that 'way' and 'path' are more similar to each other than either is, say, to 'is' or 'the', then it would be good to have the substitution of 'path'–'way' amount to less than the possible substitution 'path'–'is'. To accomplish this, a *cost* can be associated with each operation, perhaps even a different cost for each sort of insertion or substitution. Then a transformation of *minimum cost*, rather than minimum operations, can be defined. If one makes the simple assumption that a substitution costs no more than the corresponding deletion-insertion pair, then this minimum cost can be shown to obey metric properties, and defines the *generalized edit distance* between the two strings, with larger distances corresponding to less similar strings.

There is a straightforward method for computing edit distance. In a prime example of dynamic programming, the edit distance is computed for every pair of initial substrings of the two strings under study, with results for shorter substrings combining to give results for longer substrings.

More explicitly, let our two strings be $A = (a_0, a_1, \ldots, a_m)$ and $B = (b_0, b_1, \ldots, b_n)$, where a_i is the i^{th} token in string A, starting with token 1. We let the first component of the the string, a_0, be a ***null token***, representing an empty position into which we can insert.

Define also the initial substring $A_i = (a_0, a_1, \ldots, a_i)$ of a string to be the first i tokens, including the null token at the beginning.

The computation starts by assigning $D(A_0, B_0)) = 0$, the cost of transforming a_0 to b_0, the null token to itself.

$A \longrightarrow$

$B \downarrow$

	–	the	path	that	is	the	path
–	0	1	2	3	4	5	6
the	1	0	1	2	3	4	5
way	2	1	2	3	4	5	6
that	3	2	3	2	3	4	5
is	4	3	4	3	2	3	4
not	5	4	5	4	3	4	4
the	6	5	6	5	4	3	4
way	7	6	7	6	5	4	5

Figure 3: Dynamic programming for edit distance ('–' is the null token)

Each subsequent step in the computation proceeds with the simple rule:

$$D(A_i, B_j) = \min \begin{cases} D(A_i, B_{j-1}) + D_{\text{insert}}(b_j) \\ D(A_{i-1}, B_j) + D_{\text{insert}}(a_i) \\ D(A_{i-1}, B_{j-1}) + D_{\text{substitute}}(a_i, b_j) \end{cases}$$

where $D_{\text{insert}}(x)$ is the cost for inserting x, and $D_{\text{substitute}}(x, y)$ is the cost of substituting x for y.

Starting with $D(0,0)$, one can fill each $D(i,j)$ in a table, ending at $D(m,n)$, the edit distance between the two strings. The table is filled from upper left to lower right, as each entry is computed from its upper, leftward, and diagonal neighbors using the minimum rule above. Figure 3 gives this table for the example strings.

8.1. String alignments

As a by-product of the edit distance computation, one can create an *alignment* of the two strings. This alignment matches the elements of the two sequences in linear order and shows the correspondence between tokens and substrings of the two matched strings. An alignment can be generated directly from the table created in the edit distance computation by following the path of minima chosen during the computation from the upper left corner to the lower right. Rightward travel along this path corresponds to insertion of a token from string A, downward travel to tokens from string B, and diagonal paths to substitutions. (Multiple minimum paths may result, giving alternate but equivalent alignments.)

The alignment created from our two example strings (figure 4) gives the correspondence between the tokens of the two initial strings. From the figure, it is easy to see the structural similarities of the two strings.

Alignments can be created for sets of more than two

–	the	path	that	is	–	the	path
–	the	way	that	is	not	the	way

Figure 4: A string alignment table

strings. These can be expressed in terms of extended alignment tables, with added rows corresponding to the additional strings. These alignment tables could further be abstracted to probabilistic descriptions of the sequences, using either a zero-order or Markov chain description. Chan and Wang ([3]) have used *syntheses*, zero-order probabilistic descriptions of alignment tables in order to generalize the edit distance and capture the notion of distance between two sets of sequences. Techniques such as this may prove useful in later work.

9. Context Clustering

In keeping with a straightforward approach to this preliminary work, a simple clustering technique was chosen to produce hierarchical sets of keyword contexts with similar structural properties. In this approach, contexts judged most similar in terms of a generalized edit distance were grouped into clusters. This technique is similar to some methods used in automatic grammar induction ([14]).

Clustering was chosen over grammar induction or other abstract techniques for the simple reason that the result is more easily explained from the data. The resultant groupings indicate exactly which data contribute, and alignments or syntheses can help to determine the exact nature of the contribution. Grammar induction techniques give results so far abstracted from the data that analysis is often unclear.

The clustering procedure used was the group average method, a variety of agglomerative hierarchical clustering often used in biological and genetic studies in numerical taxonomy ([1]). The technique is *agglomerative* in that groups of increasing size are built from smaller groups. It is *hierarchical* in that each the members of a cluster retain their pre-existing cluster organization, as opposed to a flat structure in which the origins of cluster members are not retained.

The hierarchy produced by the clustering algorithm is useful in judging similarity in a variety of ways. Comparing the clusters at one similarity level with those groups either above or below in the hierarchy gives a good indication of which properties are responsible for the indicated level similarity. Properties of the data may become apparent due to their uniform presence (or absence) at a given level in the hierarchy.

9.1. Locality in the edit distance

There is a degree to which purely configurational (syntactic) considerations are local in nature. Syntactic well-formedness and syntactic interactions are properties and behaviors that seem to have a high locality of effect. The presence of phrasal constituents in almost every syntactic theory is evidence of the degree to which this belief is held — phrasal boundaries mark the limits of local syntactic interactions for most word classes. Only some word classes, such as verbs and event-denoting nouns, seem to affect the placement and configuration of more distant constituents. Most word types seem to affect (and, conversely, are affected by) primarily the configuration in their immediate vicinity.

In order to highlight the locality of these configurational effects, the edit distance used in the experiments was modified so as to decrease the importance of token distance from the keyword. One would like to weight near tokens more heavily, but without ignoring the contributions of distant ones. A window function (sometimes called a step function) would be simplest, but would only count near tokens and completely discount far ones. A linear dropoff function would be able to include contributions of all tokens, but because some strings are very long, it would necessitate a slow dropoff if even the very distant tokens were to contribute to the measure.

In the end, a geometrically decreasing weight function was chosen, due to its useful properties:

- Near tokens are weighted more heavily than far tokens.
- All tokens in the string still contribute to the distance measure.
- A *half power* distance can be defined, which helps in the understanding and analysis of the results.

The half power distance is the distance for which the tokens on one side (those near the keyword) account for half of the total possible edit distance, while those on the other side (farther from the keyword) account for the remainder. This helps give a more intuitive reading for the resulting distance, with an effective *window* around the keyword which can be treated equally with the remainder of the string.

The implementation of this geometric dropoff requires only a small change to the original dynamic programming algorithm for edit distance. The table-filling rule becomes:

$$D(A_i, B_j) = \min \begin{cases} D(A_i, B_{j-1}) + L^{i+j} \times D_{\text{insert}}(b_j) \\ D(A_{i-1}, B_j) + L^{i+j} \times D_{\text{insert}}(a_i) \\ D(A_{i-1}, B_{j-1}) + L^{i+j} \times D_{\text{substitute}}(a_i, b_j) \end{cases}$$

where L is the *locality factor*, which is defined in terms of the half power distance, P_h:

$L = \frac{1}{2}^{1/P_h}$, so that $L^{P_h} = \frac{1}{2}$.

9.2. Problem-specific weights

While it would be ideal to perform the analysis using only perfect equality of lexical items as a criterion, both the number of contexts required for useful generalization, and the immense computational cost of performing such experiments are prohibitive. In order to make the test procedures tractable in these experiments, lexical items were not treated uniformly as purely lexical tokens. The input was first divided into word classes based on standard part-of-speech classification, and edit distance costs were assigned on the basis of those classes.

The text was initially tagged using a stochastic part-of-speech tagger ([15]). The 48 tag types used were divided into 12 equivalence classes (verbs, nouns, determiners, adjectives, etc.) in order to simplify weight assignment. To give members of a given class a higher self-similarity, intra-class substitutions were assigned lower cost than inter-class substitutions. Perfect lexical equality was still accorded a cost of zero.

These particular classes were chosen on the basis of general linguistic knowledge with respect to the underlying functional aspects of the theory. It is hoped that in later analyses, untagged text can be used in the system from end to end, with context type and word classifications coming as a result of the pattern clustering scheme.

10. Context method results

The context clustering algorithm described above was run using a variety of different keywords. Two examples, *of* and *without*, are given to provide a basis of comparison with other methods in grammar induction and selectional frame acquisition. One example of a word relevant to the TIPSTER project is given, to illustrate applications of the similarity clustering technique in acquiring domain-specific lexicons.

199 occurrences of *of*, 197 of *without* and 150 of *joint* were chosen randomly from the 1988 Wall Street Journal [26], part-of-speech tagged, and clustered using the localized edit distance and the group average clustering method. The half-power distance used was 6, giving 3 tokens per string. Because of processing constraints, only the right-hand side of each lexical environment was used in the clustering. In order to achieve clusters of equal significance correlating both sides of the context, without assuming some intrinsic cross-correlation, the sample size would need to be increased dramatically.

The results of the clustering are given in two forms. *Dendogram* tree structures are shown on the final page. These diagram are presented in order to provide a relative indication of the structuring properties of the technique – to show that the clustering algorithm used provides more than a flat grouping. In these tree diagrams, the vertical scale represents the similarity of merged clusters. Two segments of the tree joined by a horizontal segment indicates the merger of two clusters whose distance corresponds to the height of the connecting segment. Higher connections correspond to greater distance (less similarity).

In addition to the dendograms, the context strings of some of the significant clusters are given as alignments in figures 5 through 13. The context strings are indexed by number, matching identical (although almost illegibly small) indices on the diagrams.

10.1. Prepositional arguments

The prepositional keyword *of* was used to test whether the method could extract general noun-phrase structure (NPs being the usual right-hand complement of *of*). Clusters representing the expected short NP patterns, such as [DET N], [DET Adj N], and [DET N-plural] were generated.

Two of the more interesting low level clusters are illustrated in figures 5 and 6. Figure 6 is a cluster which groups genetive NPs as the argument to *of*. Figure 5 illustrates phrasal delineation by punctuation, promising perhaps that the method could also derive the syntactic phrase-structuring properties and conventional uses of punctuation.

Another test was run with the prepositional keyword *without* , again to test the for NP structure, and to illustrate semantic subtyping of the arguments. Most of the argument clusters found were phrases denoting an event or action, either with a nominal event head (figure 7), or with a participial phrase (figure 8).

The clustering for *without* also revealed as significant the idiomatic expression 'without admitting or denying X,'

121:	of the	asahan	authority	,
96:	of the		dealer	,
136:	of the gross	national	product	,
84:	of the old		one	–
63:	of the proposed		actions	,

Figure 5: **of**: the [MOD] NOUN DELIMITER

where X is a term carrying negative connotations (figure 9).

10.2. Domain-specific vocabulary: joint

A trial using an exemplary word from the TIPSTER domain was also run, to test whether the method could extract paradigmatic use carrying semantic information. The word *joint* was selected because of its semantic relatedness to the cooperative nature of the business tie-up events (the domain of the TIPSTER task), and because of its observed heavy use in relevant context. 150 occurrences of *joint* were taken randomly from the same corpus, and clustered using the same techniques as for *of* and *without*.

The simplest clusters for *joint* are of the form 'joint X', where X is a group behavior or a group (figure 10). This kind of semantic collocation information can also be derived through statistical bi-gram analysis ([7, 18]).

The phrasal clusters produced by the method, however, cannot be obtained with bi-gram methods. Figures 11,12 and 13 illustrate clusters of paradigmatic usage of *joint* in the business reporting domain. These clusters reflect the semantic collocations that can be expected to appear with *joint*. The appearance of these clusters shows that the paradigmatic use embodied by the LCP is derivable by purely structural lexical methods.

The more structured clusters shown here for *joint* (figures 11, 12, and 13) give patterns with direct applicability to IE systems. In fact, these patterns were derived previously through other techniques and are currently used in the DIDEROT system to trigger extraction of joint venture events.

11. Conclusion

This paper has presented a linguistic framework in which to view the use of pattern-based extraction systems for

85:	of the code	's	spirit
47:	of the dollar	's recent	rise
146:	of the company	's quarterly	dividend
182:	of the president-elect	's favorite	phrases

Figure 6: **of**: the N's NP

119:	without a	significant	correction	
38:	without a	significant	retreat	
19:	without a	proper	hearing	
11:	without a	legislative	vote	
42:	without a		bone	
168:	without any		coattails	
155:	without any		results	
61:	without any		authorization	whatsoever
36:	without any	congressional	authorization	
136:	without any	prior regulatory	approval	

Figure 7: **without**: [a\|any] EVENT-NOMINAL

93:	without raising		tax	rates
7:	without raising			taxes
138:	without hurting			customers
166:	without telling			them
4:	without recognizing			it
120:	without borrowing			money
192:	without using		installment	notes
186:	without taking		a	strike
20:	without fomenting		a	revolution
134:	without complying		with federal	disclosure
47:	without	putting	up any	cash
16:	without	buying	any	shares
150:	without ever	entering	the	courthouse
159:	without bail	pending	a	hearing

Figure 8: **without**: Xing Y

100:	joint bid
91:	joint bid
115:	joint effort
104:	joint appearances
105:	joint appearance
146:	joint ventures
145:	joint ventures
140:	joint ventures
117:	joint ventures
52:	joint venture
34:	joint venture
38:	joint ventures
39:	joint ventures
50:	joint chiefs

Figure 10: **joint**: cooperative

122:	joint venture of enron	corp and sonat	inc
26:	joint venture of sammis	corp and transamerica	corp
124:	joint venture of general motors	corp and allied-signal	inc

Figure 11: **joint**: venture of X CORP and Y INC

text understanding. The framework is based on the functional aspects of denotational lexical semantics, treating the lexical and semantic components of an expression as mutual constraining parts, each imposing constraints on the structure of the other.

The viewpoint leads to an investigation of the lexical-semantic interaction in terms of a classification of structural properties. The two ends of the spectrum can be analyzed separately, bringing independent structural classifications to bear on the analysis of the interaction.

Methods were outlined for creating classifications of this sort, to create hierarchical descriptions of context and predicate types, which form a descriptive vocabulary for analyzing the interaction of lexical and semantic properties in use.

Experiments were performed on structural clustering of lexical context, using a localized edit distance as a measure of similarity. These experiments showed that structure clustering can derive the lexical information required for constructing LCPs.

Future directions: Obviously, the current level of these techniques is not sufficient to automatically create patterns mapping lexical structure to semantic denotations. What they do show, however, is that edit-distance clustering is a useful technique for extracting the syntactic portions of such patterns – from a set of less than 200 contexts in each case we see significant clusters, identical to patterns used in an existing IE system. Further work is needed in order to fold the semantic mapping into the clustering process. Metrics are needed for classifying both semantic structure and for the integrated mappings. One solution might be to augment the string edit distance with a predicate-similarity metric based on tree-matching, with the relational structure treated as a tree of predicates and arguments. This combined metric could provide a measure of similarity for classifying the structural mappings themselves.

Much of the community has discussed the need for semantically marked text, much like that in the example of figure 2, over which to run machine learning methods such as these. A collection of text with relations explicitly marked out would provide an ideal set of learning examples for the clustering technique shown, and for extension into methods integrating the semantic and syntactic clustering.

Because of the cost in analysis time, the creation of such a collection is currently unreasonable. In parallel research, I am constructing tools to allow the researcher to easily mark text relative to an arbitrary RR knowledge representation scheme.

The similarity measure could benefit from further research. As it is given, the edit distance provides no distinction between contiguous substring matches and arbitrary subsequence matching. A measurement for *reversals* – the alternation of a pair AB with BA, for tokens (or substrings) A and B – would be useful, as this sort of swapping is common in natural language. There have been some attempts toward this in the genetics community, but no significant success has been achieved.

The metric also could benefit from more advanced methods of comparing *sets* of strings, rather than pairs only. In order for these techniques to be most effective in deriving lexical structure, the comparison metric should give credit explicitly to those substructures responsible for the assessed similarity. The present metric can only do this on a pair-wise basis. The *syntheses* presented in [3] provide one method for extension to sets of strings, probabilistic grammars another.

The method also suffers from its computational complexity. The clustering method is $O(n^2)$, where n is the number of contexts, while the edit-distance computations are $O(k^2)$, where k is the average context length, making the entire method $O(k^2n^2)$. The context length, k is relatively fixed, but the number of separate contexts n is unbounded. For large numbers of context strings, the computational cost is prohibitive. However, there is a simple parallel reduction of the clustering which brings the cost down to a tractable $O(nk^2)$ for n processors. I have begun to experiment with this algorithm on a CM-5 parallel computer.

References related to edit distance and context evaluation, primarily from the biological literature, are continually coming to my attention. Unfortunately, I have not had ample opportunity to judge their relation to the present work.

Acknowledgements: I would like to thank James Pustejovsky and Paul Buitelaar for useful discussions in developing this material. I would also like to thank the anonymous reviewers for their helpful comments in its improvement.

195:	without	admitting	or	denying		wrongdoing
59:	without	admitting	or	denying		guilt
123:	without	admitting	or	denying	any	wrongdoing
194:	without	admitting	or	denying		wrongdoing
122:	without	admitting	or	denying	the	allegations

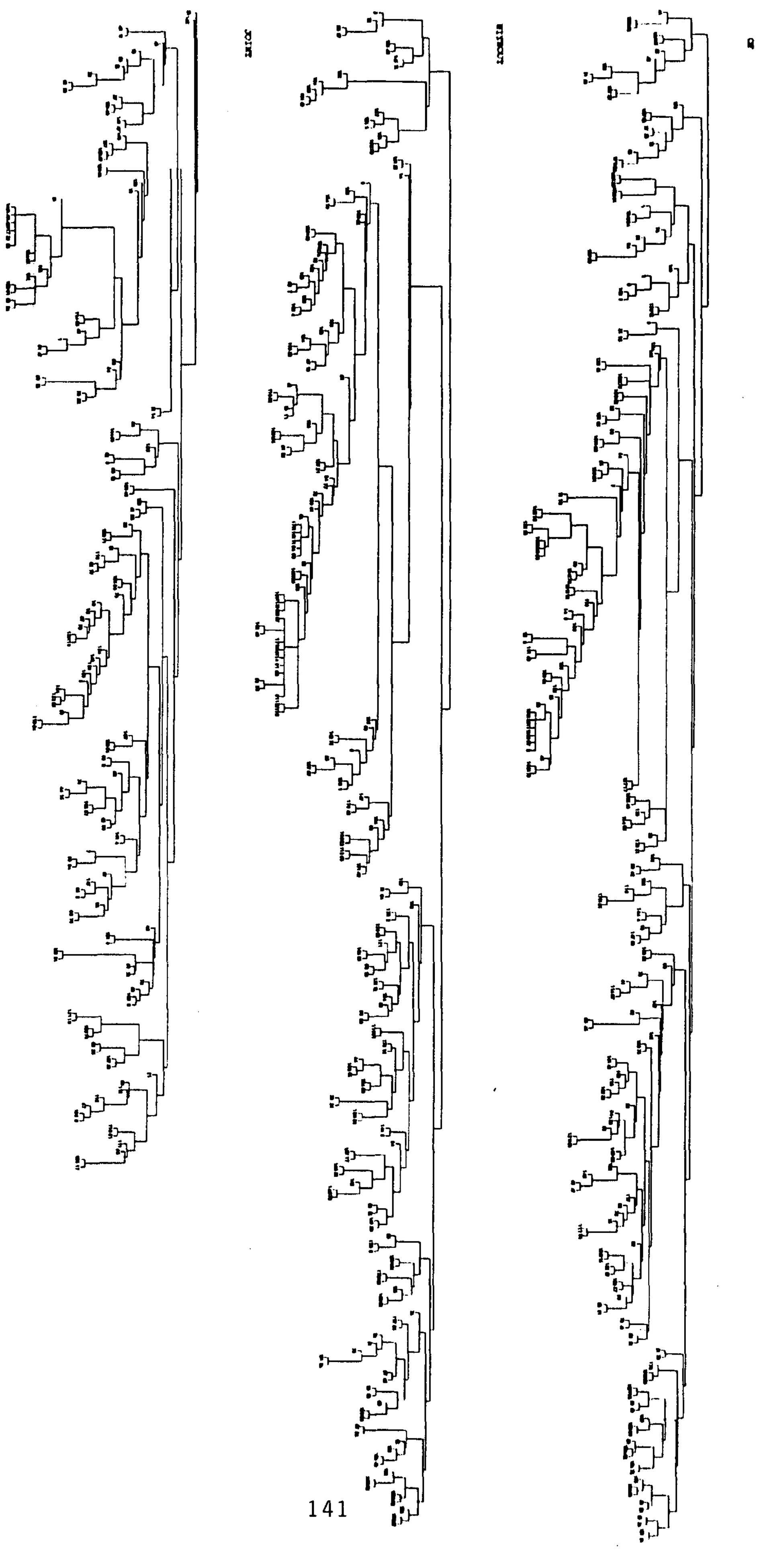
Figure 9: **without**: admitting or denying X

131:	joint	venture		with	bp	america		inc
119:	joint	venture		with	icn	pharmaceuticals		inc
73:	joint	venture		with	aaa	development		corp
67:	joint	venture		with	komori	printing	machinery	co
10:	joint	venture	agreement	with	pt	astra	international	inc
16:	joint	venture		with	french	publisher	hachette	sa

Figure 12: **joint**: venture with X INC.

113:	joint	venture	of	dow	chemical	co	,	detroit	,			and	corning	glass	works		corning	,	n	y
64:	joint	venture	of	dow	chemical	co	in	midland	,	mich	,	and	corning	glass	works	in	corning	,	n	y
98:	joint	venture	of	landmark	land	corp	,	carmel	,	calif	,	and	ranieri	wilson	co				new	york

Figure 13: **joint**: venture of X CO. LOCATION$_X$ and Y CO. LOCATION$_Y$



References

1. Anderberg, M.R. (1973) *Cluster Analysis for Applications*, Academic Press, New York
2. Boguraev, B. and Briscoe, T., Eds. (1989) *Computational Lexicography for Natural Language Processing*, Longman, London.
3. Chan, S.C., and Wang, A.K.C. (1991) "Synthesis and recognition of sequences," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **13**-12 pp. 1245-1255
4. Cowie, J., Guthrie, L., Pustejovsky, J., Wakao, T., Wang, J., and Waterman, S. (1993) "The Diderot Information Extraction System," to appear in *Proc. First PACLING Conference*, Vancouver.
5. Grishman, R., Macleod, C., and Sterling, J. (1992) "New York University: Description of the PROTEUS System as Used for MUC-4," in *Fourth Message Understanding Conference (MUC-4)*, Morgan Kaufmann Publishers, San Mateo
6. Grishman, R., Sterling, J. (1992) "Acquisition of selectional patterns," in *COLING 92*, the Proceeding of the 14th International Conf. on Computational Linguistics, Nantes, France
7. Hindle, D., and Rooth, M. (1991) "Structural Ambiguity and Lexical Relations," *Proceedings* of the 29th Annual Meeting of the ACL.
8. Hobbs, J.R., Appelt, D., Tyson, M., Bear, J., and Isreal, D. (1992) "SRI International: Description of the FASTUS System used for MUC-4," in *Fourth Message Understanding Conference (MUC-4)*, Morgan Kaufmann Publishers, San Mateo
9. Hobbs, J., Stickel, M., Appelt, D., and Paul, M. (1990) "Interpretation as Abduction," SRI International AI Center *Technical Note 449*, Palo Alto
10. Hogeweg, P., and Hesper, B. (1984) "The alignment of sets of sequences and the construction of phyletic trees: An integrated method," *J. Molecular Evolution*, **20** pp. 175-186
11. Lehnert, W., Cardie, C., Fisher, D., Riloff, E., and Williams, R. (1991) "Description of the CIRCUS System as Used for MUC-3," in *Third Message Understanding Conference (MUC-3)*, San Diego, pp. 223-233
12. Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., and Soderland, S. "University of Massachusetts: MUC-4 Test Results and Analysis," in *Fourth Message Understanding Conference (MUC-4)*, Morgan Kaufmann Publishers, San Mateo
13. Levenshtein, V.I. (1966) "Binary codes capable of correcting deletions, insertions, and reversals." *Cybernetics and Control Theory* **10**-8 pp. 707-710; Russian original (1965) *Doklady Akademii Nauk SSR* **163**-4 pp. 845-848
14. Lu, S.Y., and Fu, K.S. (1977) "A clustering procedure for syntactic patterns," *IEEE Trans. on Systems, Man, and Cybernetics*, Oct. 1977.
15. Meteer, M., Schwartz,R., and Weischedel,R. (1991) "Empirical Studies in Part of Speech Labelling," *Proc. of the 4th DARPA Workshop on Speech and Natural Language*, Morgan Kaufman Publishers, San Mateo, pp. 331-336
16. Pustejovsky, J. (1991) "The Generative Lexicon," *Computational Linguistics*, 17-4.
17. Pustejovsky, J. (forthcoming) *The Generative Lexicon: A Theory of Computational Lexical Semantics*, MIT Press, Cambridge.
18. Pustejovsky, J. (1992) "The Acquisition of Lexical Semantic Knowledge From Large Corpora," in *Proceedings of the Fifth DARPA Workshop on Speech & Natural Language*
19. Pustejovsky, J. and Anick, P. (1988) "The Semantic Interpretation of Nominals," *Proc. of the 12th International Conference on Computational Linguistics*, Budapest.
20. Pustejovsky, J. and Boguraev, B. (1993) Lexical knowledge representation and natural language processing. *Artificial Intelligence*, 1993.
21. Sankoff, D., and Kruskal, J.B., eds. (1983) *Time warps, string edits, and macromolecules*, Addison-Wesley, Reading, MA
22. Sellers, P.H. (1974) "An algorithm for the distance between two finite sequences," J. Comb. Thy **A16** pp. 253-258
23. Smadja, F. (1989) "Macrocoding the Lexicon with Cooccurrence Knowledge," First Int'l Language Acquisition Workshop, *IJCAI 89*
24. Wagner, R.A., and Fischer, M.J. (1974) "The string-to-string correction problem," J. ACM **21** pp. 168-173
25. Wilks, Y., Fass, D., Gou, C.M., McDonald, J.E., Plate, T., and Slator, B.M. (1990) "Providing Machine Tractable Dictionary Tools," *Machine Translation* **5**
26. *The Wall Street Journal*, (1988) Dow Jones, Inc.
27. Zernik, U. (1990) "Lexical Acquisition: Where is the Semantics?" *Machine Translation* **5**, pp. 155-174