1 Distinguished Usage

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

To appear in *Corpus Processing for Lexical Acquisition*, B. Boguraev and J. Pustejovsky, eds., 1994 MIT Press.

Abstract

As the various readings of the title hint, ambiguity in natural language is a pervasive phenomenon and remains a problem for human and computer understanding. This chapter presents an automatic method for distinguishing the various modes of use of a term by categorizing context cues. The technique is presented as a lexical acquisition technique for *pattern based* understanding systems. Recently, implementors of NLP systems have been moving away from using conventional brittle syntactic parsing methods, instead adopting a variety of pattern based approaches for complex language understanding 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 the lack of a framework to connect them with linguistic theory. In combining functional views of both denotational semantics and syntactic structure, this chapter provides a basis for examining the structural constraints controlling the interaction between text and denotation. An approach for determining and exploiting these structural constraints is outlined in terms of building hierarchical lexical classifications based on the contextual features found in use, and direct applications are shown for text understanding. Experiment results for such a method are given, demonstrating the functionality of the approach.

1.1 Introduction.

Recently, some implementors of NLP systems have been moving away from using conventional syntactic parsing methods, instead adopting a variety of *pattern based* approaches for complex understanding tasks such as the Message Understanding Conferences (MUC-4, 1992; MUC-5, 1993) and the ARPA-sponsored TIPSTER project. These pattern-based systems are characterized by short and fairly specific lexically indexed patterns which are used to specify the relation between strings in the source text and particular entries in a problem-dependent knowledge representation. The patterns are variously represented by regular expression strings (Cowie et al., 1993), small sets of limited-context grammar rules (Lehnert et al., 1991), finite-state automata (Hobbs & Appelt, 1992), and other suitable schemes.

This one-level process or pattern matching 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 (Lehnert et al., 1991; Hobbs & Appelt, 1992)), these systems achieve performance comparable to more standard systems that rely heavily on full syntactic analysis (Hobbs et al., 1990; Grishman et al., 1992). 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 (Lehnert et al., 1991; Cowie et al., 1993; Hobbs & Appelt, 1992), while system coverage of the text material 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 this tedious hand analysis is Lehnert's work (Lehnert et al., 1992), 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 used are weak in the sense that they cannot identify new phenomena that may have linguistic or practical relevance. Further, they use only sparse linguistic information, and provide no means of generalizing patterns across semantic domains.

1.1.1 A theoretical framework

By relating lexical pattern-based approaches to a lexical semantic framework, such as the Generative Lexicon theory (Pustejovsky, 1991), my aim is to provide a basis through which pattern-based understanding systems can be understood in more conventional linguistic terms. Within such a framework, methods for pattern acquisition can be studied and developed, while the effectiveness of patterns, in terms of functionality and agreement with linguistic theory, can be assessed.

My main contention is that such a framework can be developed by viewing the lexical patterns as structural mappings from text to denotation in a compositional lexical semantics, blurring 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, and from their theoretical relation to syntactic properties. In the framework I present, patterns indexed to individual words relate semantic interpretations to lexical constraints, in a manner dictated by the lexical items in context and their configuration. The interpretation of large constructions is made possible by matching the mutual constraints of multiple words as they appear together. This interpretive mechanism could be realized as a lexically indexed semantic grammar, implemented in a parsing formalism such as LTAGs (Schabes et al., 1988), link grammars or slot grammars. In such a system, the interpretations of single words are dictated by the constraints of context, and these constraints are combined for multiple words to provide a consistent interpretation for phrasal and sentential constructions.

1.1.2 Pattern acquisition

The approach to lexical pattern acquisition outlined here seeks to automatically separate the various modes in which a lexical item is used. These *modes of use* are distinguished on the basis of context cues — the presence (or absence) of particular lexical items and members of syntactic classes correlated with their various configurations in the context of the term under study. The distinguishing cues for these modes are defined *inductively* from a large set of example usage of the term. Because the distinctions are based on syntactic behavior theoretically related to semantic variance, these modes of use should also provide semantically relevant distinctions.

Once this sort of automatic classification of modes has been performed, it is up to the linguistically knowledgable researcher or system user to relate each to the practical semantic representation. Because the acquisition process is necessarily disconnected from the real world semantics of the language, this sort of interpretive labeling must be made in a semantically grounded manner (i.e. by the researcher). However, because a relevant classification has already been made by automatic means, the task is much simplified. If we divide lexicon development into stages of identifying relevant syntactic and semantic categories, classifying the lexical items, and attaching to each item its appropriate semantics, the first two have been performed automatically. The third, I would argue, can not be automated without a truly intelligent machine; the in-depth world knowledge required to provide the semantic interpretation of language is available only to an agent operating in the world.

Related work describing automatic lexical acquisition rarely attempts to focus on identifying the multiple uses of a single term, instead only classifying words against other words (Schütze, 1993; Basili et al., 1993; Pereira et al., 1993). Other work, primarily dealing with acquiring subcategorization frames (Brent, 1991; Ushioda et al., 1993) is similar in flavor to the present work, but searches for occurrences of a fixed set of previously identified syntactic patterns, rather than inducing them from the textual structure.

These and other works describing corpus techniques for deducing lexical structure (Pustejovsky, 1992; Pustejovsky & Anick, 1988) and semantically marked selectional frames (Grishman & Sterling, 1992) also suggest that lexical/semantic patterns can be induced from corpora, but do not directly apply to the generation of distributed pattern-based systems.

1.2 Information Extraction

The recent Message Understanding Conferences (MUCs) and ARPA TIPSTER project have posed a complex and fairly specific problem in text understanding. The problem given is that of creating semantic *templates* or *frames* to correspond to newswire and newspaper articles about a particular subject. The expressiveness of the templates is restricted and somewhat skeletal, capturing the bare facts of the text, and not its complete meaning. Hobbs (Hobbs & Appelt, 1992) 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, while other non-pertinent information may be ignored.

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 extraction 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 database object implementation) which rigidly define what types of facts and relations from the text are representable.

1.2.1 Mapping — Extraction : 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. One needs to find the 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 through a pronominal reference. In a strict ER model, two atoms, each denoting this direct object, would have to be used, 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. The important extension from the more standard model is that everything is accessible by reference, and that atomic structures are not singularly bound into one relation.

1.3 Functionality in 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. If one takes the stand that the surface structure of language is not arbitrary, but rather depends heavily on semantic constraints stemming from the concepts and relations that are conveyed through its use, then the form of these concepts becomes highly important. An accurate model of these relations and constraints will have a benefit in implying and constraining both their appearance in the language, and the possible styles of semantic representation.

Many, if not most, theories used in computational linguistics today assume some degree of *functionality* in language, with corresponding structures at the syntactic and semantic levels. This assumption is so pervasive that it can be difficult to separate the syntactic and semantic descriptions. Even purely syntactic theories exhibit this phenomenon: words are said to act as operators, or to take arguments, when what is really meant is that they appear in particular syntactic constructs. For example, a syntactic theory such as CCG (Steedman, 1986) uses terms like 'function application' and 'function combination' to refer to the behavior of lexical items, whereas the theory is a proof-theoretic analysis of syntax. The only function application available on theoretical grounds here is at a metasyntactic-level — that of applying combinators to derive categories from other categories. By positing functional behavior to the lexical items and categories themselves, one implicitly attributes meaning to them.

These functional concepts are not directly applicable to the symbols used in a language, but apply only in a semantic domain, a world of ideas and relations. This vocabulary of functional structure has its most recent origins in mathematical descriptions of meaning, such as predicate logic and the lambda calculus. That fact that accepted grammatical theories of language (e.g. CFG, LFG, CCG, HPSG, GB) have co-opted this vocabulary, and assume a parallel functional structure, should be clear evidence that these functional semantic structures have parallels in syntax.

1.3.1 Semantic constraints on language

If semantic constraints play such a role in determining syntactic properties, through what mechanisms do they operate? This is really one of the fundamental questions of linguistics, and one in which a enormous amount of effort has been expended. With the deference due to such an enterprise, I will attempt to outline what I see as the three major sorts of semantic mapping mechanism that have been explored: categorical, morphological, and structural.

All three mechanisms assume some degree of functionality in the treatment of composition; that is, terms can have semantic roles associated with them that are expected to take fillers from the surrounding context. The roles here are objects or relations that are semantically associated with the lexical entry — semantic constituents that can be integrated with the meaning of the term. An object reference such as 'chair' implies the concept of sitting, and of a sitter. 'Cooperation' implies the notion of participant, which itself implies an event to participate in. A verb such as 'eat' has associated with it an eater and something being eaten. It is obvious that not all of these associations are always realized lexically. It is the manner and degree to which these associated roles are and can be expressed that is controlled by the three following constraint mechanisms.

Categorical constraints specify the semantic or functional class of the arguments for a lexical entry. In order to fill the role, the argument must satisfy these class restrictions. In our simple 'eat' example, the filler of the **eater** role must be capable of eating. In existing lexical systems, this information, when given for verbs, is often expressed as the role labels of subcategorization frames. The type of the argument itself can be expressed in a number of ways; e.g., in a feature-based class system, it might be **+animate** or **+consumer** or some such equivalent.

This kind of categorical constraint is not limited to verbs; other forms project semantic category constraints as well. Adverbs, loosely speaking, are a class that take events as arguments. Many nouns take relational arguments as well, although it may be less obvious. *Brother*, for example, needs a person to be the brother of, as in *my brother*, or *Joe's brother*.

The category information of the role fillers needed to match these constraints can be present in a number of forms. It may be found implicitly in the semantic denotation of the filler (as with *Joe*), or it can be explicitly provided through morphology (see below) or through *functional typing*. This sort of category identification moderates role filling through secondary functional terms, which accept the ultimate filler and force a particular interpretation of it. In English, the prepositions are used primarily in this capacity, taking nominal arguments and *coercing* them to match the roles provided by higher level structure. For instance, directional terms such as *west* or *left* can take a reference argument, which is logically a location. Terms often used as fillers for this argument, however, don't have a usual location interpretation. Their interpretation is moderated by the role assignment and implicit type assignment behavior of the prepositions: 'west *of* the house,' 'left *from* the tree.' This behavior is part of a larger family of type shifting mechanisms, called *type coercion* (Pustejovsky, 1991), in which the expected or available role types force a particular interpretation of the filler.

Morphological constraints specify the relation between the available realizations of a lexical entry and the various possible subsets of its meaning. While the core meaning of an entry is fixed, certain common properties, e.g. number, time reference, etc., are more variable. These peripheral semantic properties can be fixed through the choice of morphological form.

Morphological constraints also play a large part in determining what role in some relation a lexical item is allowed to fill. In this *case marking*, morphology is used to indicate how this lexical item relates to the semantic roles being required by a higher level structure.

Morphological variations span a wide range of consistency: they can be relatively uniform across the lexicon, as in the case of tense and number variation, or can be completely idiosyncratic.

Structural constraints determine the ordering and configurational structure of the arguments. These constraints determine how the realizations of the roles of the relation denoted by a lexical item may be appear relative to one another and to the entry. Global word ordering (S-V-O, S-O-V, etc.) is a high level manifestation of these constraints. Some terms have more individual constraints: the genetive marker *'s* in English takes its single argument to the left; the two arguments to a conjunction appear to either side. No entry seems to constrain the realizations of its arguments to a particular numbers of words, so we may assume that this configurational specificity does not work at the level of the lexical unit, but rather at the level of the semantic unit. Phrasal items, whose semantics have been encapsulated as a single concept, are the items over which the configurations are specified.

None of these constraint mechanisms is inviolable; nor are they independent. The interactions between the three often displays strange variability, especially where they interact heavily. In English, for example, there is little morphological marking of roles, and structural and categorical constraints seem to bear the burden of relational assignment. However, when role assignments are made explicitly by functional type casting, as when prepositional phrases are used as arguments, positional constraints can be overridden. In a heavily case-marked language such as Latin, however, we see far less dependence on the structural constraints, as the role/filler correspondence has already been made through morphological constraints, and word order can be more variable.

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, are both subsets of the general relational models). In fact, it is often the case that the relations of the models 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 leave unexplored.)

1.4 Integrating syntactic with semantic constraints

Because these three type of interpretation constraints interact so heavily, a mechanism that explicitly examines the degree of specificity in argument *position* and in argument *type*, and especially their interaction with one another in use, should be better able to achieve the goals of interpretation; that is, to relate the text to a particular denotation.

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 applied only to verbs, some theories, such as the generative lexicon theory (GL), extend the idea to include *all* lexical categories (Pustejovsky, 1991). For the purposes of this discussion, we can consider the GL lexicon to have a three-part representation of these lexical constraints associated with every term:

1. **Semantic structure** which provides the relational structure denoted by the entry. This includes categorical constraints for fillers, and default values for unspecified fillers. These categorical constraints and relational structure are contained in the *qualia* description of the entry. 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.
2. **Appearance information** which indicates how this entry can appear at the symbol level, including morphology.
3. **Cospecification** which relates the possible fillers of the semantic structure to various realization forms, including argument configuration and syntactic realization information. These constraints can be specified much like regular expressions, and can provide varying degrees of 'fit' to the syntax.

It is easy to see how a theoretical approach such as this, based on these lexical constraints, could be operationalized in an understanding system. Using a lexically indexed system, the configurational constraints (cospecifications) of each term could be expanded against the context of that term in order to find a structural match. This match, which is essentially a parse of the context, would then be tested for consistency with the semantic constraints, to find whether there is an interpretation of each of the structural components matching the appropriate semantic type constraints. Perhaps various alternate configurations would have to be tested against the various possible interpretations of the term, but once this is done, a complete interpretation of the word in context has been formed.

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 (Cowie et al., 1993).

1.5 Patterns

Pattern-based understanding systems combine syntactic and semantic processing through the use of *patterns*. Patterns consist of lexically indexed syntactic templates that are matched to text, in much the same way as are regular expressions, along with type constraints on substrings of the match. Portions of the syntactic template are referenced to the semantic representation of the lexical item, so that text found matching these portions of the pattern must also have denotations consistent with the corresponding semantic roles.

These patterns can be thought of as 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. In general, the pattern matching system is a local phenomenon, which doesn't seek to create a global syntactic analysis. However, because of typing constraints interacting between multiple patterns, global structures and long-distance phenomena can be observed.

1.5.1 DIDEROT, a pattern example

For example, in the DIDEROT project (Cowie et al., 1993), a pattern is represented as a GL structure (GLS) which gives the syntactic context along with mappings from text variables to a 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 1.1.

The cospecification information is contained here 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. Other parts of the `cospec` are either literals, such as 'venture' or 'agreement', which must match the text exactly, or variables (A1, A2, A3...). The `args` field indicates how argument variables should be realized syntactically. `A1` and `A2` here must be `type(np)`, where `np` designates a heuristic class of noun phrases.

```
gls(establish,
  syn(...),
  args([arg1(A1,
       syn([type(np)]),
       qualia([formal([code_2,organization])])),
     arg2(A2,
       syn([type(np)]),
       qualia([formal([code_2,joint_organ])])),
     arg3(A3,
       syn([type(np),morph([establish,etablished,...])]),
       qualia([formal([code_2,organization])]))]),
  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],
     [A2,was,self,with,A3]]),
 types(tie_up_verb),
 template_semantics(pt_tie_up,
     tie_up([A1,A3],A2,_,existing,_))).
```

Figure 1.1
A GLS for 'establish'

The last element of the cospec, '`*`', is a Kleene star over all tokens — anything or nothing may appear in this position.

The semantic structure of the entry is given in the `template_semantics` field, which specifically relates the arguments which match the appearance template (the `cospec`) to the relational semantic of the knowledge representation. The argument variables are restricted to a semantic *type path*, such as `[code_2,joint_organ]`, given in the `qualia` field. The type path establishes a region in a type hierarchy which must contain the type of the argument (Pustejovsky & Boguraev, 1993).

Because of the difficulty and expense of deriving patterns, GLSs could not be produced for every term of importance in DIDEROT. 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 subgrammars 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.

1.6 The current state of Pattern Acquisition

The TIPSTER and MUC projects have provided a wealth of knowledge about the difficulty of 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 interpreted by the project sponsors), one can, by careful reading of the text, usually find those segments of text which correspond 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.

At least two of the systems used in the MUC and TIPSTER projects used some automatic acquisition techniques. The *Autoslog* component of the CIRCUS system (Lehnert et al., 1992) attempted to find valid patterns from a pre-parsed input, using an inference mechanism derived by matching new text against previously filled templates. Even though the patterns are produced from single occurrences of matches, so that any one pattern is unlikely to generalize, the system achieved surprising performance through shear coverage. This system reduced the pattern lexicon development time by hundreds of ours.

In the DIDEROT system, two levels of automatic pattern construction were used. First, information from structured dictionary entries (Boguraev & Briscoe, 1989; Wilks et al., 1990) was used to derive moderately effective, but fairly unspecific lexical patterns. Further pattern acquisition was performed by automating concordance studies to find syntactic patterns which were statistically relevant over many uses of a term. The techniques presented here are an outgrowth of this work on inducing patterns from examples found in corpus.

1.7 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 any linguistic structure, used in a particular fashion will give rise to the same denotation today as it will tomorrow. 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 focuses 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.

The reason for this focus stems from the nature of the available data. Raw text is abundant – consider the hundreds of megabytes of text available through newswire services, through email and USENET, or through reference corpora. This raw text gives direct information about the *structural* constraints a term is subject to — how it appears relative to other terms. It does not offer information as readily for semantic or morphological forms of lexical knowledge. To build a database of sense tagged, or even simply phrase-bracketed text requires enormous amounts of manpower not available to the average lexical researcher. This considerations limit us to investigating only the structural aspects of the corpus data.

One method for investigating these structural interactions is that of classifying their uses into categories which have some *structural similarity.* This similarity can be defined either in syntactic or semantic terms. In terms of the function-argument structure or reference-relation representations, words with similar type ambiguities 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 their functional properties within the language (number and typical realization of arguments) might differ considerably.

This work provides a step in forming a vocabulary of categorical (semantic) and structural (syntactic) classes through which we can describe the constraints acting between words. Classification methods for both lexical and semantic structure are outlined here. As I have already mentioned, however, semantic data is hard to come by, and it would be more practical to limit the study to the lexical structure only. An experimental implementation of the lexical approach is presented in the latter sections of the paper.

1.7.1 Acquiring 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 pat-

$$\cdots \begin{bmatrix} p(\alpha_2{=}\mathrm{w}_1) \\ p(\alpha_2{=}\mathrm{w}_2) \\ \vdots \\ p(\alpha_2{=}\mathrm{w}_3) \end{bmatrix} \begin{bmatrix} p(\alpha_1{=}\mathrm{w}_1) \\ p(\alpha_1{=}\mathrm{w}_2) \\ \vdots \\ p(\alpha_1{=}\mathrm{w}_3) \end{bmatrix} \; k \; \begin{bmatrix} p(\beta_1{=}\mathrm{w}_1) \\ p(\beta_1{=}\mathrm{w}_2) \\ \vdots \\ p(\beta_1{=}\mathrm{w}_3) \end{bmatrix} \begin{bmatrix} p(\beta_2{=}\mathrm{w}_1) \\ p(\beta_2{=}\mathrm{w}_2) \\ \vdots \\ p(\beta_2{=}\mathrm{w}_3) \end{bmatrix} \cdots \tag{1.7.1}$$

Figure 1.2
A simple probabilistic representation of the context about k.

tern systems achieve their performance by relying on the expected structural properties of the language. If one can capture 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, then one has a description of its structural constraints associated with it. This is, of course, what traditional systems do with grammar rules which dictate what structures, and with what other terms, a word may be used. Using a lexicalized grammar helps to make this relationship explicit, but it exists in all grammars.

From the acquisition point of view, however, we are presented with a very lexicalized version of the data. To study a word's usage, all we need do is observe the literal context in which it is used. Of course, some suitable representation of that context is needed. Partial parse trees, as in LTAGs, would be perhaps ideal, but the data we are presented with contains no parse or bracketing information.

Instead, we can form a description using the data we have — the identity and position of the surrounding lexical items. Using this, we can form a probabilistic description of what structure we expect to find in context. Given enough data, we might be able to form distinct categories of what that context looks like.

For example, the simplest probabilistic description of a context would have, for each position preceding and following the item, a distribution of the likelihood of occurrence of every other word (see fig. 1.2). This is very similar in style to n-gram models, which are used along with Markov models of language (see e.g. (Baker, 1975)). The primary difference here is that n-gram models are primarily used to predict a single item of a sequence after having observed $n-1$ items, whereas a context description such as this is used to predict the n items surrounding the one of primary interest.

A lexicon which used these context vectors to encode the structural constraints would have many context vectors for each entry. Each vector would capture one of the *modes of use* of the term — one of the principle categories of context in which it is used. Since the structural constraints of an entry are dictated by its meaning, the term would have a separate interpretation in each of the different modes. In the pattern lexicon presented earlier (fig. 1.1), these separate modes of use are represented crudely by regular expressions in the `cospec` field. Each mode is given a separate mapping from its components to the relational semantic structure of the entry.

To acquire such a lexicon, a collection of usage would be analyzed in order to pick out these natural classes of context. Since they are defined purely in terms of the lexical environment, the collection would have to be fairly large in order to observe enough occurrences of similar structure to ensure statistical relevance.

These 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 more general semantic properties as well as those which are idiosyncratic to the individual entries.

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.

1.7.2 Semantic structure

In an analogous way, if denotations were available for a sufficiently large quantity of text, the relational structure of the denotations, as it relates to the text could be classified purely from their structural properties. This would give a method for inducing the categorical constraints that go hand-in-hand with the structural constraints given above, but also for inducing the relational structure of the denotations themselves. In exactly the same manner as for context classes, relation predicates could be grouped hierarchically based on similarity of 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 certainly not as widely 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 allow, as with lexical structure, to identify properties which are generic to all denotation, and those which are idiosyncratically related to individual terms. It would also provide a mechanism for creating a hierarchy of predicate types, form those which are very specific in their structure and categorical constraints, to those which are more loosely constrained and adaptable to circumstance.

1.7.3 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 understanding systems, and of how such a structural acquisition system might operate, consider the following example, from the TIPSTER/MUC-5 domain:

Imagine a researcher developing the domain-dependent vocabulary for an extraction 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. 1.3). 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.

Now that the researcher has provided a connection between text and denotation, the system can use the classifications of context and relation types as a vocabulary to describe the structural mapping. For instance, it is now known that 'IBM', and also 'Motorola', which may have been unknown terms, can act as AGENT arguments, and specifically, as the AGENT arguments of a DEVELOPMENT predicate. The system is also given the information that "devices" can act as a PRODUCT within certain relationships, as can "tools".

This annotated text also informs the system that the phrase "is jointly developing" can sit within this particular structured context of phrasal units, and that it bears a certain relationship to the DEVELOPMENT predicate. If this relationship can be correlated with other examples of "developing" mediating a similar context to a DEVELOPMENT predicate, then the system can infer a more general mapping between "developing" within this sort of context and the DEVELOPMENT predicate.

Assuming the system has already acquired a domain-independent encoding of the co-agentive functionality of 'with', now there is evidence specifically that

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

DEVELOPMENT$_A$
- AGENT:
  - "IBM"$_B$
  - "Motorola"$_F$
- PRODUCT:
  - "tools"$_C$
    - TYPE:
      - X-RAY
    - USE:
      - MANUFACTURE$_D$
        - PRODUCT:
          - "devices"$_G$
            - FEATURE_SIZE:
              - 0.25 μM$_E$

Figure 1.3
A segment of text, marked against a predicate interpretation

the DEVELOPMENT predicate allows this behavior, and that a configuration giving that interpretation can be generalized as:

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

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

Once the system has been provided with enough text-denotation pairs particular to the domain, it may, relying more heavily on inferred structural knowledge, postulate a general mapping for the word "is", relating the syntactic pattern

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

to a predicate structure such as:

X-PRED
- ARG_1
- ARG_2

(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.

1.8 Lexical Clustering using Edit Distance

In order to create a catalog of lexical structural descriptions, one needs a method for comparing and classifying the strings of words that make up the contexts. One method for judging the similarity between strings of lexical items (tokens) is the *edit distance* formulated by Levenshtein (Levenshtein, 1966). 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 (Sellers, 1974; Wagner & Fisher, 1974) have been used on a variety of problems in biology, genetics, speech and handwriting analysis (Sankoff & Kruskal, 1983), as well as in syntactic analysis of formal languages (Lu & Fu, 1977). (For a good introduction with applications to many domains, see (Sankoff & Kruskal, 1983).)

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.

$A \longrightarrow$

$B \downarrow$

	–	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 1.4
Dynamic programming for edit distance ('–' is the null token)

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. 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 1.4 gives this table for the example strings.

1.8.1 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 1.8.1) 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.

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

Figure 1.5
A string alignment table

Alignments can be created for sets of more than two 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, like that in figure 1.2. This type of representation has also been used for describing and classifying typical sequences of base-pairs in RNA sequences (Chan & Wang, 1991).

1.9 Context Clustering

A simple clustering technique was chosen which would produce a hierarchical classification of 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 (Lu & Fu, 1977).

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 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 (Anderberg, 1973). 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.

1.9.1 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 $L^{P_h} = \frac{1}{2}$.

1.9.2 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 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 (Meteer et al., 1991). 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.

1.10 Context method results

The context clustering algorithm described above was run using a variety of different lexical items. 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 (WSJ, 1988), part-of-speech tagged, and clustered using the localized edit distance and the group average clustering method. The half-power distance used was 6. Because of processing constraints, only the right-hand side of each lexical environment was used in the

of	the	asahan	authority	,
of	the		dealer	,
of	the gross	national	product	,
of	the old		one	–
of	the proposed		actions	,

Figure 1.6
of: the [MOD] NOUN DELIMITER

of	the code	's	spirit
of	the dollar	's recent	rise
of	the company	's quarterly	dividend
of	the president-elect	's favorite	phrases

Figure 1.7
of: the N's NP

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. (even though *can* safely assume such cross-correlation, it is even safer to study each side independently before making such claims.)

The results shown are alignment tables of the highest-similarity significant clusters formed in each case. Although the clustering algorithm produces a hierarchical categorization, the flat-grouping tables are easier to examine in this limited space.

1.10.1 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 1.6 and 1.7. Figure 1.7 is a cluster which groups genetive NPs as the argument to *of*. Figure 1.6 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 1.8), or with a participial phrase (figure 1.9).

The clustering for *without* also revealed as significant the idiomatic expression 'without admitting or denying X,' where X is a term carrying negative connotations (figure 1.10).

without	a significant	correction		
without	a significant	retreat		
without	a proper	hearing		
without	a legislative	vote		
without	a	bone		

without	any			coattails	
without	any			results	
without	any			authorization	whatsoever
without	any		congressional	authorization	
without	any	prior	regulatory	approval	

Figure 1.8
without: [a|any] EVENT-NOMINAL

without	raising	tax	rates
without	raising		taxes
without	hurting		customers
without	telling		them
without	recognizing		it
without	borrowing		money
without	using	installment	notes

without	taking	a		strike
without	fomenting	a		revolution
without	complying	with	federal	disclosure

without		putting	up	any	cash
without		buying		any	shares
without	ever	entering		the	courthouse
without	bail	pending		a	hearing

Figure 1.9
without: Xing Y

1.10.2 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

without	admitting	or	denying		wrongdoing
without	admitting	or	denying	any	wrongdoing
without	admitting	or	denying		guilt
without	admitting	or	denying	the	allegations

Figure 1.10
without: admitting or denying X

joint	bid
joint	effort
joint	appearances
joint	appearance
joint	ventures
joint	venture
joint	chiefs

Figure 1.11
joint: cooperative

joint	venture of enron	corp and sonat	inc
joint	venture of sammis	corp and transamerica	corp
joint	venture of general motors	corp and allied-signal	inc

Figure 1.12
joint: venture of X CORP and Y INC

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 1.11). This kind of semantic collocation information can also be derived through statistical bi-gram analysis (Hindle & Rooth, 1991; Pustejovsky, 1992).

The phrasal clusters produced by the method, however, cannot be obtained with bi-gram methods. Figures 1.12,1.13 and 1.14 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 such paradigmatic use is derivable by purely structural lexical methods.

The more structured clusters shown here for *joint* (figures 1.12, 1.13, and 1.14) give patterns with direct applicability to information extraction 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.

1.11 Conclusion

This paper has presented a linguistic framework in which to view the use of pattern-based extraction systems for 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, along with the form of the source data, leads to an investigation of the lexical-semantic interaction in terms of a classification of lexical structural properties. The two ends of the spectrum can be analyzed separately, bringing independent structural classifications to bear on the analysis

joint	venture		with	bp	america		inc
joint	venture		with	icn	pharmaceuticals		inc
joint	venture		with	aaa	development		corp
joint	venture		with	komori	printing	machinery	co
joint	venture	agreement	with	pt	astra	international	inc
joint	venture		with	french	publisher	hachette	sa

Figure 1.13
joint: venture with X INC.

joint	venture	of	dow	chemical	co	,	detroit	,	
joint	venture	of	dow	chemical	co	in	midland	,	mich ,
joint	venture	of	landmark	land	corp	,	carmel	,	calif ,

and	corning	glass	works		corning	,	n	y
and	corning	glass	works	in	corning	,	n	y
and	ranieri	wilson	co				new	york

Figure 1.14
joint: venture of X CO. LOCATION_x and Y CO. LOCATION_Y

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 identifying and characterizing diverse modes of use.

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 extraction 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 1.3, 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, however, the creation of such a collection is currently unreasonable.

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.

Acknowledgments: I would like to thank the editors for their helpful comments on this work, as well as the attendees at the SIGLEX workshop at which it was first presented. This work was funded in part under ARPA contract #MDA904-91-C-9328.

Anderberg, M. R. (1973). *Cluster Analysis for Applications.* Academic Press, New York. originally Anerbergs thesis.

Baker, J. K. (1975). Stochastic modeling for automatic speech understanding. In Reddy, D. R. (Ed.), *Speech Recognition.* Academic Press.

Basili, R., Pazienza, M. T., & Velardi, P. (1993). Hierarchical clustering of verbs. In *Acquisition of Lexical Knowledge from Text, Proceedings of the SIGLEX Workshop.*

Boguraev, B. & Briscoe, T. (Eds.) (1989). *Computational Lexicography for Natural Language Processing.* Longman, London. gidlines to using Longman's LDOCE for lexical acquisition.

Brent, M. R. (1991). Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29^{th} Annual Meeting of the ACL.*

Chan, S. & Wang, A. (1991). Synthesis and recognition of sequences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(12):1245–1255.

Cowie, J., Guthrie, L., Pustejovsky, J., Wakao, T., Wang, J., & Waterman, S. (1993). The diderot information extraction system. In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, Vancouver.

Grishman, R., Macleod, C., & Sterling, J. (1992). New york university: Description of the proteus system as used for muc-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4).*

Grishman, R. & Sterling, J. (1992). Acquisition of selectional patterns. In *Proceedings of the 14^{th} International Conference on Computational Linguistics.*

Hindle, D. & Rooth, M. (1991). Structual ambigity and lexical relations. In *Proceedings of the 29^{th} Annual Meeting of the ACL.*

Hobbs, J. & Appelt, D. (1992). Sri international: Description of the fastus system used for muc-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4).*

Hobbs, J., Appelt, D., & Pal, M. (1990). Interpretation as abduction. Technical Report Technical Note 449, SRI International AI Center, Palo Alto.

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., & Soderland, S. (1992). University of massachusetts: Muc-4 test results and analysis. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*.

Lehnert, W., Cardie, C., Fisher, D., Riloff, E., & Williams, R. (1991). University of massachussetts: Muc-3 test results and analysis. In *Third Message Understanding Conference (MUC-3)*, pp. 223–233.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and control Theory*, 10(8):707–710.

Lu, S. & Fu, K. S. (1977). A clustering procedure for syntactic patterns. *IEEE Trans. on Systems, Man, and Cybernetics*.

Meteer, M., Schartz, R., & Weischedel, R. (1991). Empirical studies in part of speech labelling. In *Proc. of the 4th DARPA Workshop on Speech and Natural Language*, pp. 331–336.

MUC-4 (1992). *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, San Mateo. Morgan Kaufman.

MUC-5 (1993). *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, San Mateo. Morgan Kaufman.

Pereira, F., Tishby, N., & Lee, L. (1993). Distribution clustering of english words. In *Proceedings of the 31^{st} Annual Meeting of the ACL*.

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics*, 17(4).

Pustejovsky, J. (1992). The acquisition of lexical semantic knowledge from large corpora. In *Proceedings of the Fifth DARPA Workshop on Speech & Natural Language*.

Pustejovsky, J. & Anick, P. (1988). The semantic interpretation of nominals. In Proc. of the 12^{th} International Conference on Computational Linguistics.

Pustejovsky, J. & Boguraev, B. (1993). Lexical knowledge representation and natural language processing. *Artificial Intelligence*.

Sankoff, D. & Kruskal, J. (Eds.) (1983). *Time warps, string edits, and macromolecules*. Addison Wesley, Reading, MA.

Schabes, Y., Abeillé, A., & Joshi, A. K. (1988). Parsing strategies with 'lexicalized' grammars: Applications to tree-adjoining grammars. In *Proceedings of the 12th International conference on Computational Linguistics (COLING-88)*, Budapest.

Schütze, H. (1993). Part of speech induction from scratch. In *Proceedings of the 31st Annual Meeting of the ACL.*

Sellers, P. (1974). An algorithm for the distance between two finite sequences. *J. Comb. Thy*, A16:253–258.

Steedman, M. (1986). Combinators and grammars. In Oehrle, R., Bach, E., & Wheeler, D. (Eds.), *Categorical Grammars and Natural Language Structures.* Foris.

Ushioda, A., Evans, D. A., Gibson, T., & Waibel, A. (1993). The automatic acquisition of frequencies of verb subcategorization frames from tagged corpora. In *Acquisition of Lexical Knowledge from Text, Proceedings of the SIGLEX Workshop.*

Wagner, R. & Fisher, M. (1974). The string-to-string correction problem. *J. ACM*, 21:168–173.

Waterman, S. A. (1993). Structural methods for lexical/semantic patterns. In *Acquisition of Lexical Knowledge from Text, Proceedings of the SIGLEX Workshop.*

Wilks, Y., Fass, D., Gou, C., McDonald, J., Plate, T., & Slator, B. M. (1990). Providing machine tractable dictionary tools. *Machine Translation*, 5.

WSJ (1988). The wall street journal.