Matching Texts to Extract Information

Kenneth B. Haase

13.1 Introduction

This chapter describes an implemented approach to broad coverage semantic extraction from natural language texts. Semantic extraction, the determination of semantic roles for textual elements, is important to any processing or presentation tasks requiring identification of structures within and among documents, including summarization, question answering, or categorization.

Work on analysis below the document level has typically tried for either deep understanding of a small corpus in a circumscribed domain (e.g. [46, 6]) or shallow template-based extraction from a large corpus over a circumscribed domain (e.g. [29, 25, 26]). *Broad coverage semantic* *extraction* is in this second tradition of shallow analysis, but addresses general-purpose methods and representations applicable without substantial initial domain knowledge.

The key characteristic of our approach is the use of graph matching between sentence representations to replace the application of templates to text. Query sentences are matched to target documents and target documents are matched with one another in order to obtain correspondences between textual elements. These matches are constrained by a combination of syntactic and lexical analysis where the lexical analysis makes substantial use of WordNet [33] [32] to broaden its coverage. The use of text to text matching obviates manual or automatic generation of templates for extracting particular facts from the database.

The graph matching algorithms are part of a general model of *analogical representation* [16] where the functions provided by canonical structures (e.g. case frames, conceptual dependencies, etc) are instead dynamically computed by heavily constrained (and thus tractable) graph matching. In this account, cognitive schematic structure emerges dynamically from the indexing and matching of new experiences against a rich background of past experiences. The work described here applies the analogical representation model to a database of descriptions generated by analyzing a large text corpus. In this application of the analogical representation model, individual texts become the "experiences" indexed and matched by the system.

13.1.1 The Corpus

We are currently working with a 10 million word corpus provided by Gannett Corporation. It consists of 8 years of news summaries from the periodical **USA Today**(TM). The corpus was chosen because our approach currently works best on small texts where multi-paragraph phenomena (long-distance anaphora, stage-setting, discourse, etc) seldom occur. Though understanding such phenomena is important, we wished to simplify the cases we were studying and the USA Today summaries fit these purposes perfectly.

The corpus is represented by a semantic network of over six and a half million nodes indexed by a variety of strategies. We are still in the midst of comparing different indexing approaches and doing formal evaluations of their relative performance. Because index generation is resource intensive and our indexing strategy is still provisional, the evaluations in Section 13.6 are based on a 2.2 million word subset of the entire corpus.

13.1.2 The Implementation

Our system consists of four modules:

- the parser tags text and segments it into atomic phrases
- the "grounding" module produces a "pre-semantic" network for each parsed sentence
- the "matcher" determines correspondences between pairs of these networks

- the "indexer" identifies "good" candidates for matching with a given network

Underlying the implementation is a novel infrastructure supporting:

- large (multi-million node) persistent semantic networks
- distribution of processing functions and data across multiple machines
- a fast inverted index facility supporting complex structured keys

This chapter describes each of the system's four modules in some detail. It then discusses the system's performance in two ways: a detailed examination of the system's matching of particular cases and a broader quantitative analysis of the system's performance on retrieval and extraction across the corpus as a whole.

13.2 The Parser

When we began this work in 1990, there were no robust broad-coverage parsers generally available. After searching for existing solutions, we set about implementing our own, keeping in mind the relaxed constraints of our target representation: we didn't necessarily need to generate the correct interpretation of the text as long as we could get some useful constraints on what the correct interpretations might be.

Today, there are numerous broad-coverage parsers [52, 31, 47, 5], all of whose performance is comparable to (and in some cases better than) our own. At some point, we may adopt one of these other, linguistically more sophisticated, parsers into the multi-scale framework of our current parser.

13.2.1 Parsing at Multiple Scales

The parser operates on multiple scales, using different mechanisms at different levels of processing:

1. Lexeme segmentation
2. Part of speech tagging
3. Phrasal segmentation

rather than using a single method (such as link grammars [47] or chart parsing [1]) on the input text. The multi-scale design introduces a framework where each mechanism can be extended to just the point where its reliability began to degrade. It's output would then be the input for another mechanism at the next level of processing.[1]

The structure of our parser is **not** intended as a cognitive model of human sentence processing; indeed, we believe that human understanding involves access to memory and matching much earlier in its processing. But the purpose of this parser is to construct a initial corpus to be matched against and it has served this function quite adequately.

13.2.2 Lexeme Segmentation

At the lexeme level (i), simple local rules break a string into individual sentences and then into individual words, identifying some proper names, dates, numbers, and recurring collocations as single lexical units. For instance, a sentence like "Fruit flies descend upon Los Angeles. They're expected to stay for two hundred eight-seven days." would be segmented as:

Fruit_flies descend upon Los_Angeles

They are expected to stay for two_hundred_eighty-seven days.

The sentence breaking algorithm uses a few simple heuristics and dictionary of abbreviations to avoid terminating sentences prematurely. Both sentence and word segmentation are conservative: sentences are broken apart and words merged into collocations only when the segmentation or combination is virtually certain to be correct.

Because our final representation connects to WordNet, we started by using collocations based on WordNet categories and augmented by a list of place names. However, some of WordNet's collocations had to be excluded because their constituent words had several senses only some of which could be included in a given collocation. For instance, WordNet contains the verb collocation "point to", but if this sequence is merged into a single word at the lexeme level, a sentence such as 'The Dow fell 5 points to 3333.44' would be (mis-)segmented as 'The Dow fell 5 points_to 3333.44' which would not be parsed correctly.

In addition, a few special procedures identify dates, times, and numbers, merging their individual components into single lexical items. Finally, the segmentation algorithm expands contractions using a simple lookup table; it limits this expansion to contractions which cross atomic phrase boundaries (e.g. "I've" is expaned to "I have" but "shouldn't" is not expanded to "should not").

13.2.3 Part of speech tagging

At the tagging level (ii), the segmented text is passed to a part of speech tagger. The tagging of the sentence from the previous section would be something like:

Fruit_flies$_{\mathrm{NOUN}}$ descend$_{\mathrm{VERB}}$ upon$_{\mathrm{PREP}}$ Los_Angeles$_{\mathrm{NAME}}$

They$_{\mathrm{PRON}}$ are$_{\mathrm{AUX}}$ expected$_{\mathrm{VERB}}$ to$_{\mathrm{TO}}$ stay$_{\mathrm{VERB}}$ for$_{\mathrm{PREP}}$

two_hundred_eighty-seven$_{\text{NUMBER-ADJ}}$ days$_{\text{NOUN}}$.

This tagger uses an *optimizing finite state machine* (OFSM) to assign parts of speech to the text. The optimizing finite state machine finds the lowest cost path through a network of states whose arcs are annotated with costs for particular inputs. For instance, the cost of traversing a "noun" arc for the word "dog" is substantially less than the cost of traversing a "verb" arc for the word. The costs assigned to words and arcs are based on a combination of morphological rules and sense distribution data from the Brown corpus [55].

Optimizing finite state machines implement what Charniak calls *probabilistic grammars* [3]. These, in turn, have a computational similarity to Hidden Markov Models [40, 4] with a difference only in their generation: HMMs are generated empirically, while probabilistic grammars (or OFSMs) are generated from hand-coded specifications. This increases their overall coverage (because they have rules that are not just based on patterns which happened to occur in the training corpus) at the cost of some reduced accuracy (where the assigned costs are inappropriate to the material being parsed).

The grammar uses an idiosyncratic set of tags which distinguish identical parts of speech in different phrasal roles. For instance, the category 'adverb' is subcategorized into:

- adverb modifying adjective (*adjverb*)
- adverb modifying preceding verb (*ending-adverb*)
- adverb modifying preposition (*padverb*)
- adverb modifying subsequent verb (*adverb*)
- adverb without contiguous attachment (*dangling-adverb*)

each of which is specially treated by the phrasal segmentation mechanism. This subcategorization also allows more precise control over the association of particular words with particular grammatical roles.

The part of speech tagger also recognizes certain syntactic tropes with semantic significance: for instance, phrases of the form "proper-name, proper-name" where one of the names is a known place name (e.g., "Washington, D.C.") are explicitly tagged as such; this information is then is used in the generation of the final text representation.

13.2.4 Phrasal Segmentation

The tagged text is passed to a simple algorithm for aggregating phrases based on their tagging. This algorithm divides the set of tags into "head tags" (nouns, verbs, proper names, etc) and "modifier tags" (everything else: adjectives, adverbs, prepositions, etc). A table associates each head tag with a set of pre-modifiers and a set of post-modifiers.

For instance, the pre-modifiers of verbs include auxiliaries, adverbs, the word "to", and various complementizers; the post-modifiers include adverbs and time identifers. Given this division, a phrase consists of a head word (tagged with a head tag) preceded by words tagged with pre-modifiers and succeeded by words tagged with post-modifiers.

Our current algorithm is reasonably robust (mid to high 90s on a per-word basis) but is plauged by two recurring problems: long proper names and conjunctions. Long proper names are often segmented into distinct phrases (for instance, "Bar Harbor Yacht Club" is segmented as "Bar Harbor Yacht" and "Club"). The immediate cause of this error is really at the tagging level, which identifies "Yacht" and "Club" as distinct proper names. We are looking at resolving these problems either in lexeme segmentation or in some post processing step.

Conjunctions are complicated because they can create "atomic phrases" out of smaller entities which could readily be atomic phrases of their own. At the tagging level, we identify modifier conjunctions (e.g. 'the sweet and sour sauce' or 'he walked slowly and carefully'), leaving phrase-level conjunctions to be resolved (or not) at the text representation level.

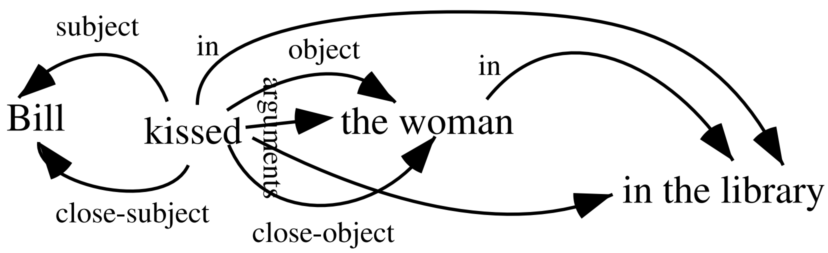

Figure 13.1
The arcs in the figure indicate micro-relations generated by our parser for the sentence "Bill kissed the woman in the library".

13.3 The Text Representation

The result of the parsing process is transformed into a semantic network intended to support analogical matching. The nodes in the network represent phrases in the text and are connected by two kinds of relations: *micro-relations*[2] (which encode grammatical structure, e.g. Figure 13.1) and *association relations* (which encode word meaning and lexical structure, e.g. Figure 13.2). The generation of this representation is called "grounding" because it involves the connection of terms to the shared ontology of word meanings and relations.

Micro-relations describe grammatical and thematic connections between phrases, especially subject-object relations (regularizing for the passive construction) and preposition attachment. Each preposition is translated into a micro-relation and an inverse micro-relation. Both preposition attachments and subject/object relations are over-generated, yielding some spurious relations. These spurious relations are often (see Section 13.6) resolved by the context of matching, though more accurate linkings would definitely increase the system's accuracy in both matching and indexing. Following [51], the preposition "of" is especially handled with connection to the preceding phrase (except in the case of "accused of" forms). Figure 13.1 shows the micro-relational structure for the sentence "Bill kissed the woman in the library".

The basic subject-object and preposition relations are combined into several derived relations. For verbs, preposition attachments and object relations are generalized into an `arguments` relation connecting an

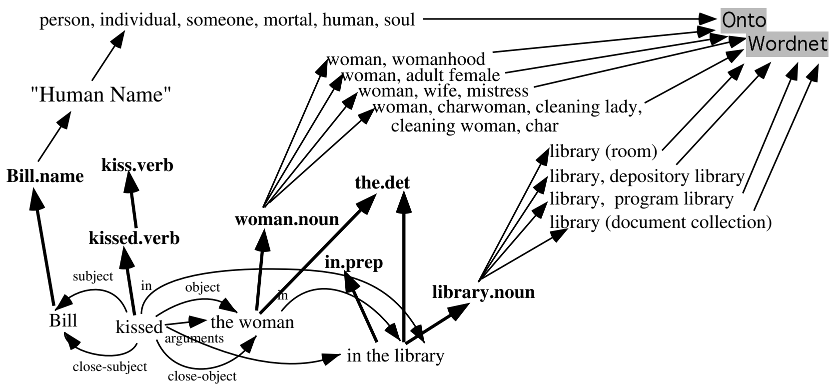

Figure 13.2
The associations of the entire sentence are also either rhizome entries (in **bold**) or concepts in the ontology derived from WordNet (in *italics*). We only show the first set of associations in WordNet.

atomic verb phrase to all its possible arguments. Similarly, the overgenerated subject and object relations are specialized into single value `close-subject` and `close-object` relations indicating the possible subject and object closest to the verb. These relations are derived to support matches between networks despite some differences in grammatical role.

Association relations are generated by lexical analysis and are the foundation of the matching process: the system only considers matches between nodes that have associations in common. Association relations connect nodes to either nodes in other contexts (phrases in other sentences) or to nodes in a common ontology.

The ontology has two parts: a **rhizome**[3] in which actual words are organized by part of speech; and a **concept space** generated from the synsets and hypernym (generalization) relations of WordNet. Figure 13.2 shows how the sentence of Figure 13.1 is associated with the ontology. The phrase itself is associated with a rhizome entry (in **bold**) for every constituent word. The rhizome entries are associated with their possible senses in the concept space. These concepts all correspond to WordNet synsets, which are each associated (in turn) with a node corresponding to their hypernym.

Rhizome entries are distinguished from each other based on spelling, part of speech, and some simple semantic heuristics. For instance, the noun, verb, adjective, and adverb senses of the same word have distinct rhizome entries. Proper names are heuristically categorized as "agents" (people, organizations, companies, etc.) and "locations" (cities, states, countries, etc). Thus, the representation distinguishes "Clinton" (the person) from "Clinton" (the city). When the system cannot distinguish whether a name refers to an agent or a location, the corresponding phrase is associated with both possible interpretations.

Figure 13.2 shows the association relations in the rhizome and WordNet for the sentence of Figure 13.1. This structure is the basis for the matching algorithms described in the next section.

13.4 Matching

The system matches descriptions in two phases: first, **cognate matching** proposes a set of matches based solely on common associations; then, **structure matching** extends these matches based on micro-relational structure. While graph matching in general is combinatorially intractable, reasonable performance (worst case quadratic, average case linear) is achieved by **a)** limiting the candidate matches to those with justifying associations and **b)** returning only the unambiguous matches between the descriptions.

13.4.1 Cognate Matching

> **Definition:** *Two elements are* **cognates** *if they share some unique common association which is associated with no other elements of the descriptions being matched.*

Unlike matches based on distance between terms in a taxonomy [41], cognate matches are not sensitive to the density or sparseness of the links. Also, cognate matches are 'defeasible,' since each match has a concrete justification rather than a numerically calculated or otherwise emergent explanation. An important feature of cognate matching is its *contextual sensitivity*: two descriptive elements can be cognates in the context of one match but not in another, depending on the other elements of the context. Since we are attempting to match descriptive elements in corresponding contextual roles, this is an important prop-

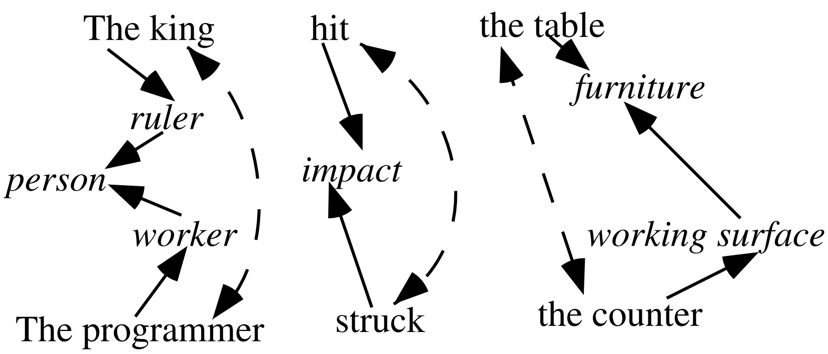

Figure 13.3
Type-unique cognates occur when the elements of two contexts have distinct and unique associations in the global ontology. In this case, distinct associations (solid lines) through a simplified ontology (italics) lead to obvious cognate relations (dotted lines).

erty.

There are two interesting special cases of the cognate relation: type unique cognates and triangle cognates. **Type unique cognates** (Figure 13.3) occur when the elements of a description are so different from each other and so similar to elements of the other description that a match can emerge solely on the basis of common associations to nodes in the global ontology. Thus, "The king hit the table" and "The programmer struck the chair" would have immediately determined cognate relations based on the within-context dissimilarity of people, actions,

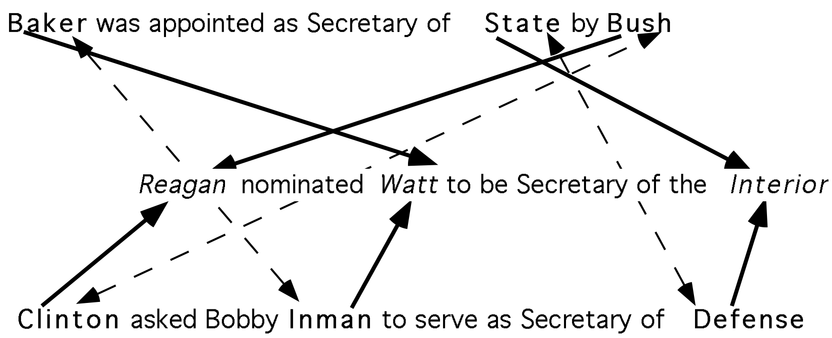

Figure 13.4
Triangle cognates occur when the elements of two contexts are associated with a third common context. Associations (solid lines) through distinct elements (in italics) of this third context determine cognate relations (dotted lines) between elements of the two contexts (in bold). For clarity, we have removed the cognate relations reflecting word identity or synonymy.

and furniture.

Triangle cognates (Figure 13.4) become possible when nodes are associated with nodes in other descriptions. In a triangle cognate, the previous association of two descritpions with a third description enables cognate relations through the elements of the third description.

Ridiculous Matches Cognate matching can readily produce ludicrous matches, for instance, between 'Mary waited on the table with the nasty patrons' and 'Mary sat on the chair with the ugly seat':

$$Mary \iff Sally$$

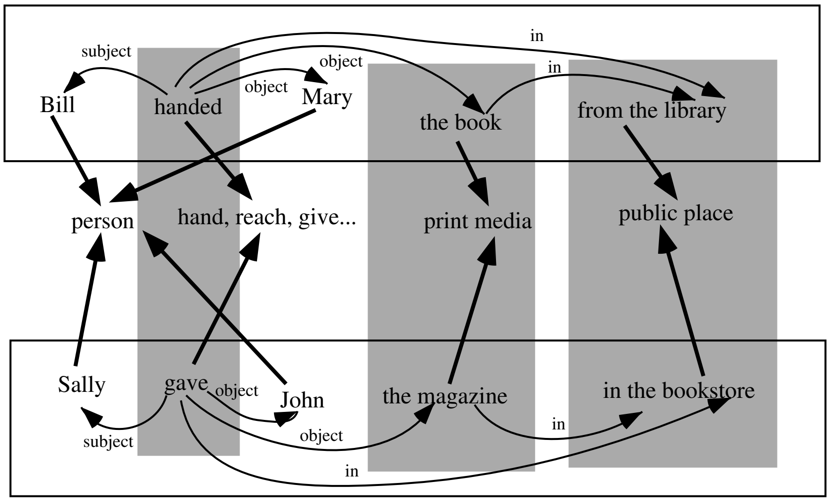

Figure 13.5
In this situation, cognate matching cannot determine how the human individuals are related to one another, since they're all associated with the prototype *person* (in an ontology simplified for clarity). The grey boxes indicate the matches which the system *is* able to determine by cognate identification alone.

$$sat \iff waited$$

$$chair \iff table$$

$$seat \iff patrons$$

Because of this, the correctness of semantic extraction also relies on the indexing mechanism, which proposes and ranks descriptions (documents) to be matched.

13.4.2 Structure Matching

Cognate matching will fail to generate a match between two nodes when either:

1. they have no associations in common, or
2. they have no associations *uniquely* in common

When the associational structure of the representation is as rich as the structure provided by WordNet, the first case seldom occurs. However, the second case is quite common, occurring whenever elements of a description are similar enough to one another (have the same associations) to produce an ambiguity. Structural matching uses micro-relational structure to narrow the context of the match so as to resolve this ambiguity.

Figure 13.5 shows networks describing the sentences:

- Bill handed Mary the book from the library

- Sally gave John the magazine in the bookstore

Three cognate relations are initially found: between "handed" and "gave" (based on *pass, hand, reach, ... give*) and "book" and "magazine" (based on *print media*) and between "library" and "bookstore" based on *public place*. The human actors involved in the sentences all have the same common associations (*person* and its associations) and so cannot be cognates. Structure matching, however, reduces the space of cognates to the boxed regions in Figure 13.6 where the potential ambiguity no longer exists, resulting in the correct matching of the actors to one another.

13.4.3 Limits of Structure Matching

Cognate and structure matching can only find corresponding structures in text when phrase structure is roughly similar. The handling of association and the use of relations (rather than sequencing) makes the matching approach more general than simple template matching, but it

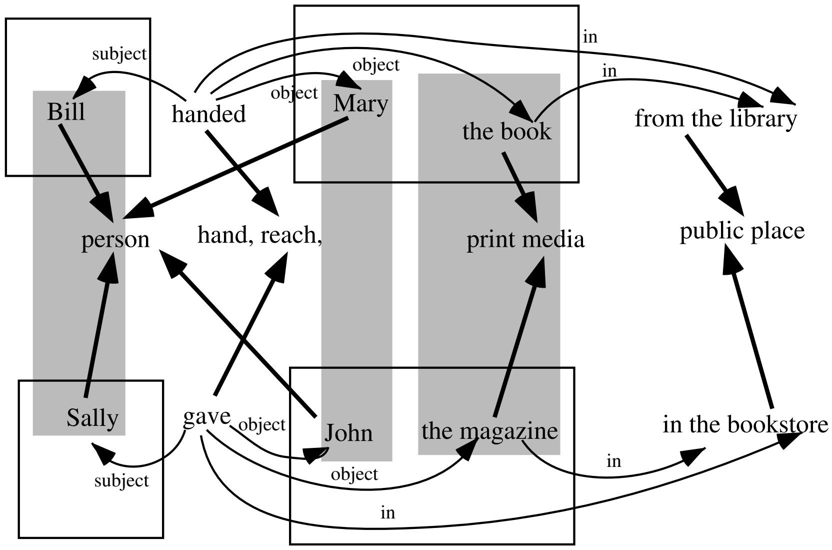

Figure 13.6
When micro-relations are used to reduce the context being considered, cognates relations can be determined. The left hand bounding box represents the reduced context determined by the `subject` micro-relation; the right hand bounding box is likewise derived from the `object` micro-relation. Within these contexts, cognate relations (the grey boxes, as above), are readily determined.

still relies on structural consistentency. Because of this, there are clearly cases where it will fail. We can divide these into three broad categories:

1. same structure, different meaning

   (e.g. 'The Redskins met the Patriots' or 'Clinton met Major')

2. different structure, same meaning

   (e.g. 'John gave Mary the book' or 'Mary received the book from John')

3. different structure, "implied" meaning

   (e.g. 'Clinton nominated Inman' and 'Clinton asked Inman to serve')

Case (i) can only be resolved by some sort of disambiguation to separate the distinct cases. This could be done on two levels: word sense disambiguation (distinguishing the senses of meet) or a more indirect "document level" disambiguation which keep texts containing radically different word senses from being matched to one another. For instance, if the indexing and retrieval mechanism can guarantee that "The Redskins met the Patriots" will only be matched with other athletic competitions, it is not necessary to disambiguate the exact sense of "met" used in the text.

Case (ii) could be helped by a better parser and lexicon capable of generating thematic roles as micro-relations, rather than relying on the grammatical structure. Note that such connections would still support ambiguity, but the 'more canonical' representation would automatically generalize cases like 'gave' and 'received'. Another solution would be to somehow learn those correspondences, which is an approach we also consider for case (iii).

Case (iii) is the hardest problem in general because there can be no 'first principles' solutions to it. The connection between 'Clinton asked Inman to serve' and 'Clinton nominated Inman' is an entailment which is a fact about the world: we cannot generate it from some linguistic first principles. We can see this by noting how a linguistically innocuous substitution of "Haase" for "Clinton" yields the dissimilar sentences

let $mark(x)$ be a function from elements to elements, initially undefined
let $in\_conflict(x)$ be a function from elements to True or False
let $taken(x)$ be a function from elements to True or False
let $hypotheses$ be a set of element triples $< x, y, z >$
let $results$ be a set of element pairs $< x, z >$
For every element e of X:
    "First, we mark all the associations of each element"
    for every association a of e
        if $(mark(a) \neq undefined)$
            if $(mark(a) \neq e)$
                then set $in\_conflict(a) = True$
            else set $mark(a) = e$
For every element e of Y:
    "We search from the elements of Y for marked, conflict free, associations."
    "Whenever we find one, we see if it is currently taken for any other e,
    declaring it in conflict if it is"
    let $taken\_this\_pass(x)$ be a boolean function, initially $False$ for all x
    for every association a of e
        if $(mark(a) \neq undefined)$ and $(in\_conflict(a) = False)$
            if $(taken(a) = True)$ and $(taken\_this\_pass(a) = False)$
                then set $in\_conflict(a) = True$
            else set $taken(a) = True$ and set $taken\_this\_pass(a) = True$
                add $< e, mark(a), a >$ to hypotheses
For every element $< x, y, z >$ in hypotheses
    if $(in\_conflict(z) = False)$
        add the pair $< x, y >$ to $results$
Finally, return $results$

Figure 13.7
This algorithm computes the cognate relations between two descriptions in $O(mn)$ time, where n is the number of elements in the two descriptions and m is the number of associations for each element.

'Haase asked Inman to serve' and 'Haase nominated Inman'.

In Section 13.9, we discuss an approach to accquiring the knowledge underlying the correct interpretations of cases like (iii) and to special-case handling of instances like (ii). The kernel of the idea is that descriptions of similar events with different phrasings and different factual revelations (for instance, 'asked to serve' .vs. 'nominated') are indexed together and associated with one another. This association enables the cognate algorithm to find matches even when there is no structural correspondence between the linguistic descriptions. Whether this mechanism will suffice is an open question which we are vigorously pursuing.

13.4.4 Algorithms

The cognate relations between two descriptions can be found $O(mn)$ time, where m is the depth of the association network and n is the number of elements in the descriptions. This is done by a marker passing algorithm [39, 9, 17, 8, 2] that iterates over the elements of each de-

```
To structure match X and Y
    let M be the cognate matches between X and Y
    for each < x, y > in M,
        for each slot s of x
            let M' be the cognate matches between s(x) and s(y)
            for each < x', y' > in M'
                if x' or y' is not currently mapped in M
                  add < x', y' > to M
    finally, return M
```

Figure 13.8
This algorithm extends the cognate relations between two descriptions by combining micro-relational constraints with associational knowledge. The worst case time for this algorithm is $O(kmn^2)$, but the average case for matching text descriptions is closer to the same $O(mn)$ bounds of cognate matching.

scription and over the associations of each element, looking for common associations between contexts but ruling out any association common within the context.

This algorithm (show in Figure 13.7) contains three separate iterations $O(mn)$ over the elements of the description and their associations, giving the $O(mn)$ bound. Cleanup time — resetting bits, for instance — has similar bounds, maintaining the overall $O(mn)$ performance.

The algorithm for structure matching uses cognate matching repeatedly as a subroutine. Shown in Figure 13.8, the algorithm starts with the cognate matches and then iterates over the corresponding slots (micro relations) of each pair of cognates. The resulting sets of connections are then passed back to the cognate algorithm to generate new relations. This algorithm is $O(kmn^2)$ in the worst case, where k is the number of different micro-relations and the n^2 comes from the inner iteration over the sets determined by micro-relations. However, in sparse graphs such as sentence representations, even ambiguous ones, this inner iteration generally involves a small constant number of nodes rather than n, giving expected performance bounds close to the $O(mn)$ of cognate identification.

13.5 Indexing

As we described above, the matcher is not designed to generate only sensible matches, but to generate the "most sensible" match for a given pair of descriptions. The requirement of "absolute" sensibility falls equally on the mechanism used to propose descriptions to match as on the matcher itself. This problem is the *indexing problem* of case based reasoning: finding past examples which can organize reasoning about the task at hand. Work in this area [45, 28] has suggested that indexing based on goals and causes is generally the most effective; unfortunately, our pre-semantic representations are innocent of such abstractions and we are stuck with a more superficial kind of indexing and retrieval.

We are currently exploring and comparing a range of indexing strategies. Their general form is based on the vector space indexing model [43], but extends the notion of a key to include items more complicated than words or word roots. In particular, we have explored and are exploring:

1. **Literal indexing** where the keys are literal word roots[4]
2. **Sense indexing** where the keys are word root/sense pairs (e.g. noun and verb senses of the same root are distinguished)

3. **Slot indexing** where the keys are $<$ relation, root/sense $>$ pairs (e.g. $<$ subject-of, meet.verb $>$)

4. **Relation indexing** where the keys are $<$ root/sense, relation, root/sense $>$ triplets (e.g. $<$ Clinton.name, subject-of, meet.verb $>$)

A detailed comparison of various strategies, including the extension of root/sense pairs to associations in WordNet, is currently underway. The evaluations reported here (in Section 13.6) are based on "slot indexing" (method 3) where each sentence is indexed by the slot-values of the words comprising it. For instance, the key $<$ `subject-of`, `meet.verb` $>$ indicates that the sentence contains some object related by a `subject-of` link to an object associated (in the technical sense) with the verb form of meet. The actual value used as a target is the rhizome entry corresponding to the word and broad part of speech.[5]

In slot indexing, a sentence like "Napoleon met Cromwell" would generate the keys:

```
<subject,Napoleon.name>      <object,Cromwell.name>
<arguments,Cromwell.name>    <object-of,meet.verb>
<close-object-of,meet.verb>  <subject-of,meet.verb>
<close-subject-of,meet.verb>
```

while a sentence like 'Clinton met with Cromwell' would generate these keys:

```
<subject,Clinton.name>       <with,Cromwell.name>
<arguments,Cromwell.name>    <subject-of,meet.verb>
<close-subject-of,meet.verb>
```

with an overlap of two keys.

These keys are then used to incrementally construct a large inverted index from keys to sentences. Sentence similarity is judged by adding together, for all the keys in common between two documents, some weight for each key. We have experimented with both using constant weights or weights based on the key's inverse document frequency (IDF) following

[43]. To our surprise, there seemed to be little difference between these in our preliminary examinations and in our evaluation, we use constant weights. One reason for this insensitivity may be that we are searching for the individual keys themselves (which we can match against) rather than for the putative topics for which the keys are indicators.

13.6 Evaluation

In evaluating our system, we start with single queries, retrieve based on the query, and then match the query with each of the retrieved texts. The mappings returned by the matching process are then evaluated for correctness against a "complete match" determined by hand and based on textual elements in corresponding functional roles. When the target text had several elements in similar functional roles, we generously gave the system credit if it found the most syntactically obvious one. For instance, in the sentence '...Mikhail Gorbachev will become the first Kremlin leader to meet the Pope," we give the system credit for identifying "Kremlin leader" as the individual meeting the Pope.

The results of our evaluation are summarized in the table below. They are based on 8 queries against an 2.2 million word subset of our larger database:

1. "Napoleon met Cromwell"
2. "Police arrested Atilla"
3. "Cromwell nominated Pepys as secretary of defense"
4. "The Dow Jones industrial average fell 5 points to 33385.9
5. "The Corinthians invaded Rome"
6. "The man was shot by an unnamed gunman"
7. "Napoleon arrived in Elba Tuesday"
8. "Macbeth killed Duncan"

For each query, the 100 top matching articles were used (in query (5), there were not 100 matching articles). None of the queries except (1) were used in developing the matcher and indexer. The small number of queries tested against the database reflects the painstaking effort required to validate the system's matches.

Query	"Facts"	Hits	Recall	Misses	Precision	Perfect
1	86	65	76%	31/25	68%/72%	79
2	159	73	46%	75/75	49%/49%	30
3	136	84	62%	131/24	39%/78%	67
4	300	295	98%	5/5	98%/98%	96
5	22	9	41%	15/12	38%/43%	6
6I	89	33	37%	124/48	21%/41%	59
7	167	107	64%	75/55	59%/66%	49
8	147	95	65%	58/53	62%/64%	54

For each text retrieved, we recorded how many matches the system should have gotten to the text ("Facts"), how many correct matches it got ("Hits") , and how many of its matches were incorrect ("Misses"). Following [35], we distinguished recall (the percentage of facts successfully found) and precision (the percentage of found facts which were actually correct). In addition, we included adjusted precision numbers (explained below) which reflect the number of errors due solely to incorrect indexing. Finally, the number of "perfect" matches (where all the "facts" were extracted) is listed in the "Perfect" column; for all but case 5, this can be read as a percentage of the hundred cases retrieved.

The system's performance represents ranges from stellar (case 4) to miserable (case 6).

The recall performance needs to be taken with a grain of salt, as it is a function of "facts" among the articles retrieved and not of the database as a whole. This may not be representative, since the retrieval was intentionally biased towards cases where the structure of the document text was the same as in the query. In particular, cases with semantic similarity but structural difference (variant phrasings) are not well represented in this measure, since the indexing algorithm was unlikely to retrieve them to be matched against. As we begin to implement mechanisms for handling these problems, it will be important to evaluate them in more comprehensive fashion.

The precision gives a measure of how much the system over-generates as a result of its retention of ambiguity and polysemy and the lack of domain knowledge in the matcher. As we mentioned previously, precision is a function of both the matcher (which produces mappings given two descriptions) and the indexer (which finds descriptions to match). Because of this synergy, we cannot learn much by evaluating the matcher alone, but we can factor out — to some degree — errors solely due to the indexer. In particular, cases with the wrong sense of the relevant verb (for instance, meet "athletically" rather than meet "politically") have 'zero facts' to extract, and we can remove them from the precision calculation to characterize how the matcher would have done if the indexer were perfect. In the table above, we present both the raw and adjusted number of misses and the consequent precision.

In our next stage of development, we will try and characterize the most common errors in the above evaluation and address them in the parsing and indexing modules of the system. We will also attempt to implement some possible solutions to the limitations discussed in Section 13.4.3. Another obvious next step would be to assess performance over some pre-encoded mono-domain textbase (such as the MUC-3 corpus and templates) to get a better handle on the actual recall numbers.

13.7 Related Work

This section describes related work in three different areas: computational models of analogy in AI and cognitive science; knowledge representation in AI; and semantic extraction from texts in AI and information retrieval. I'll consider related work in each of these areas in turn.

13.7.1 Related Work in Analogy

For purposes of discussion, I will subdivide the field of work on analogy into three broad categories:

- structure matching approaches (e.g. [12])

- connectionist approaches (e.g. [23])
- emergent approaches (e.g. [34])

I will just sketch enough about each of these categories to compare and contrast our approach. A comprehensive recent review can be found in [21].

Structure Matching Approaches The paradigmatic case of structure matching approaches is Gentner et. al.'s SME [10] and the accompanying cognitive model [12]. The earliest work in this area include [7] and [54]. A first methodological point of difference is that, unlike SME's matching of descriptions of electron flow to descriptions of fluid flow (for instance), the work described here focusses on matching descriptions which are already in the same domain with substantial similarity. This reflects our interest in analogical processing as a foundation for representation rather than in analogical reasoning as an isolated "problem solving" activity.

Despite this difference in intent, our matching method is similar to the structure matching approaches in its combination of ontology (the associational structure) and relational constraints (the micro-relational structure) to drive matching. It differs in considering a smaller space of possible matches (only those justified by common associations) and returning a single possible match (whatever is consistent). There are two reasons for these limiting assumptions: the need for efficiency in a procedure we are considering fundamental to representation; and an interest in everyday analogies between contexts starting with substantial similarity between their components.

A second and more profound difference is that previous approaches assumed that "understanding" precedes analogy, while our approach has analogy drive understanding. The structures being matched in SME and related approaches are already resolved representations; in our approach, they become resolved representations through the matching process.

Note that our indexing approach is similar to MAC/FAC [13] in doing a flat search and then matching the results of that search. We differ in that the keys we use in our flat search are not just primitive terms but include combinations of relations with primitive terms.

Of special related note is the use of analogical text matching in machine translation [27, 44]. Many of the mechanisms and representations used in this work is similar and their recourse to analogy is the same as ours: a demand for broad coverage. While they address the same matching issues as this work, their indexing problem is limited to the number of "translation examples" rather than the sort of large corpus used in our work.

Connectionist Approaches The most notable examples here are Thagard and Holyoak's ACME [23] and ARCS [50] systems.[6] ACME does analogical matching of pairs of situations: ARCS uses similar mechanisms for both selecting a candidate for a match and doing the matching itself.

Both systems build networks to represent possible matches where the nodes are element-to-element pairings and the links are excitory or inhibitory connections indicating the consistency or inconsistency (respectively) of the pairings with each other. The novel aspect of this approach is that connections are generated from both relational information and element similarity information (in ACME, derived from WordNet); relaxing the network attempts to reconcile these two sources of information. This is in contrast to both our approach and SME, where these two sources of constraint are either used independently (SME) or with a strict dependency of structural matching on similarity matching (our approach).

ACME, like our approach and unlike SME, is *contextually sensitive*: the addition of an extra element to a situation being matched may rule out certain matches. This makes the matcher more conservative, since contextually ambiguous mappings are rejected. Unlike ACME, the contextual sensitivity in our approach is *defeasible*: contextual effects are always directly traceable to particular descriptive and assocative elements. Debugging, design, and explanation are simplified when the number of causes for a success or breakdown is small, rather than the combination of many interacting causes. At some level in actual cognition, such complex systems are doubtless ubiquitous, but for design purposes, we would like this level to be as "low" as possible.

A second difference, driven by purpose more than theory, is the implementation's scalability. Relaxation algorithms are typically slow on conventional hardware and while they might fit the demands of supporting a cognitive theory, they are not a sufficient basis for an application (nor were they meant to be). The difference in purpose engenders a difference in evaluation: as a cognitive model, ACME can be readily justified by less than a dozen examples [24] indicating the model's plausibility. As a prototype retrieval system, the hundreds of matches summarized in Section 13.6 are only a provisional validation of our approach.

Work with ARCS with relatively small databases (the 125 stories discussed in [50]) has suggested that relaxation time levels off as the number of descriptions rise. However, it appears (from their descriptions of their networks), that the total number of nodes and connections in the system rises exponentially with the number of examples, since each possible mapping has to be represented as a node in the system. This makes it impractical to consider scaling ACME without substantial respecification.

Emergent Approaches Emergent approaches broadly include those deriving Douglas Hofstader's work with CopyCat and its desendants (e.g. [34], [11], or for a review [22]) which combine representational generation or evolution with the matching process. In this framework, a community of simple recognition procedures, called codelets, compete to describe two situations in ways that permit analogies between them. The overall system is complex both structurally and dynamically, but the outcome of its processing is **both** a representation of the situations **and** a mapping between those representations. Thus, the problem it addresses is harder than those addressed in the preceding approaches or in our approach. Researchers into such systems argue that this difficulty is environmentally realistic and that the assumption of *a priori* representation builds in most of the analogical structure which the systems end up discovering.

Our approach is midway between these poles: it assumes a semantic description but not a single canonical representation. In our matcher, the search for analogies is also a search for interpretations justifying such analogies. For instance, consider the representation of the sentences:

1. The asteroid hit the planet

2. The hill was hammered by rocks

The description of these sentences is ambiguous regarding the sense of 'hit' being used; it includes twelve distinct senses, including:

1. murder, slay, hit, dispatch, bump off, polish off, or remove
2. come by, find accidentally, stumble across, come across, stumble upon, hit
3. reach, attain, make, get to, hit arrive at, or gain
4. hit or cause to move by striking
5. hit, conk, or deal a blow to
6. bump, knock, or hit
7. score, hit, tally, or gain a point

but when the descriptions are matched, it is through the "reasonable" senses (4, 5, 6, and some not shown) that the two descriptions are linked. Basically, the algorithm figures out which interpretations (concepts) enable the match between the descriptions, much as Copycat's search for codelet combinations picks out the relevant features of a proposed analogy. In determining the match, the system is seeing the sentence *as* an instance of impact or contact rather than as an instance of murder or assassination. Certainly, its representational and perceptual construction is not as radical as Copycat's, but it is still significant and (we believe) environmentally plausible for everyday life where substantial existing structure constrains both action and interpretation.

13.7.2 Related Work in Knowledge Representation

This work is related to the broad area of knowledge representation in three ways. First, it involves a model of representation (analogical representation) which is substantially different from those common in the field. Secondly, it uses descriptions which adopt common representational ideas (semantic networks, taxonomic hierarchies). Finally, it uses a representation which explicitly represents ambiguity and processes which pervasively use that representation.

A detailed discussion contrasting analogical and canonical representations would not be appropriate for this context, though Section 13.9 offers a start and [16] takes it a little futher. Beyond this, the structures used in the system's descriptions are quite conventional: frame-based description [14][49], semantic networks [39], hierarchical concept ontologies [30] (derived from WordNet), and distributed algorithms [8] [2] [9] for quickly computing complex relations over them.

The chief innovation in this area is the algorithm for cognate determination, which combines the notions of similarity and context in a simple and efficient way. First, it provides a similarity metric which is not sensitive to local density of links or assignment of weights in a taxonomy. Second, it uses context — rather than any absolute definitions — to determine similarity: this allows the same mechanism to account for both distant analogies (Grass dies;Men die) and straightforward analogies (Men die; Laertes dies) while providing a preference criteria (uniqueness of common associations) for choosing among them when they compete.

Our approach to ambiguity has the primary virtue of providing a matching mechanism which handles ambiguity of both sense (lexical structure) and relation (syntactic structure). The idea that representations preserve ambiguity is not new to either natural language or cognitive science; much of current theory [15] assumes that ambiguity is carried through to at least the interpetation of the text in its context, an assumption consistent with our model. However, in our model, while matching can resolve ambiguity, there remains the possibilty that an ambiguous sentence may remain so. We would argue that for some sorts of communcations, particularly in real-life dialouges, the ambiguity is part of the meaning. This precludes the conventional model-theorietc account of ambiguity [20, 38] from being anything but an especially elegant special case.

Our lexical representation is more similar to Hirst's "Polaroid Words" [18], where words have their meanings developed as more and more constraints are placed on their intepretation. In Hirst's system, the intepretation of text involves the accumulation and eventual resolution of constraints on the interpretation of individual words and their phrasal attachements. In our system, we attempt to simply represent these constraints in the pre-semantic representation and then resolve them when matching in context. The "world knowledge" used to constraint meaning in Hirst's system would be replaced, in our system, by disambiguated examples indicating valid interpretations.

[19] extends the "Polaroid Word" model to a "mixed-depth" semantic representation where different parts of a description have differing degrees of precision, allowing the integration of partial and ambiguous descriptions much as in the approach described here.

The significant difference between Hirst's approach and ours is that we intentionally overgenerate senses and relations, rather than accumulating constraints on interpretation. The disadvantage of this overgeneration is the generation of spurious matches. However, the advantage of the approach described here is that it does not demand that linguistic processing produce a consistent, conflict-free set of constraints to accumulate.

13.7.3 Related Work in Text Extraction

Since the MUC (Message Understanding Conference) competitions [35, 36, 37] began in 1989, there has been steady and striking progress in the extraction of facts and other information from documents. The most successful approaches [29][26] generally rely on some sort of template matching over shallowly parsed texts. While the templates are domain-dependent, recent work has explored the semi-automatic generation of such templates from a large corpus [42, 48].

This work replaces the applications of template to text with the matching of grammatical/thematic graphs to one another. The advantage of this is that it uses the natural structure of the corpus to organize it and thus can be fully automated; the disadvantage is that it loses some of the precision of hand-filtered or coded templates. Furthermore, some manual intervention is required in the face of phrasal and pragmatic variations (Section 13.4.3).

We have not yet compared our system's performance to the other MUC systems, though we hope to do so in the near future. The performance in our evaluation over the USA Today corpus indicates that its performance is promising but uneven. However, this unevenness needs to be evaluated in the light of the fact that the system was not designed for the queries or domain it was given.

13.8 Future Work

The work described in this chapter naturally leads in three directions: improving the system's performance as an extraction system, building applications based on the extraction mechanism, and deepening the system's understanding of the texts its processes,

We will be looking at performance improvements in several areas: speeding up the text processing (we hope to gain an order of magnitude over our current 2000 words/minute LISP-based performance), accquiring matching knowledge based on annotated corpora (e.g. MUC), and a more comprehensive comparison of the system's performance to domain-dependent extraction systems.

We also plan to integrate the extraction module into several applications, including a personalized newspaper and a semi-automatic email response system. The latter system would match new messages to past messages and automatically reword replies to those past messages so as to refer to the particulars of the new message. This process would normally produce several possible replies (based on past responses to the similar messages) from which a single reply could be then be selected, edited, and sent.

An approach to the deepening of the system's knowledge, with some possible objections, is discussed in the next section.

13.9 But Does it Understand?

What lies between text extraction and text understanding? Both our approach and template matching are too skeletal to be considered full-blooded understanding systems. In this section, I propose an architecture for a text understanding system which is a successor to the system described in the preceding sections. Most of the pieces of this architecture are already available: what remains is their integration and evaluation in a sufficiently challenging environment.

13.9.1 The Index and Match architecture

In this architecture, texts are parsed and analyzed much as this chapter describes, possibly using a more linguistically sophisticated parser to identify possible thematic roles (instead of possible preposition attachments) and somewhat reduce the ambiguity of the generated descriptions. This process may also do a little more with handling anaphora and trying to merge textual references.

Given this generated description, the architecture then searches for texts with similar structure to the new text and attempts to match them with it. The results of this matching are installed as associations of the new text's elements with elements of existing texts. When this has been done, the description is indexed as in the current system, but now the keys are elements of past texts rather than words or word senses.

Occasionally, the system encounters "forced matches" which associate elements of one context with another in ways which cognate and structure matching would not normally determine. These forced matches, which might be provided by a human tutor or forced by circumstance, are installed as associations and their contexts are indexed. As a result of this indexing, future texts with similar structure will be associated with these contexts and — by association — with the other context involved in the forced match.

[16] describes an example of this kind of matching and indexing mechanism which learns the passive form for particular verbs. The forced alignment of two sentences like “Socrates convinced Aristotle” and “Aristotle was convinced by Socrates” causes future passive forms of “convinced” to be correctly interpreted with respect to future active forms.

This index and match architecture allows the system to accquire rules for non-structural correspondences by being shown direct examples. In essence, the system is learning to “see similarities” by being shown similarities. The generality of these similarity judgements is still an open question, but they certainly extend the scope of the system beyond simple structure matching.

13.10 Summary and Conclusions

This chapter described a text extraction system which uses text-to-text matching as the basis for extraction. The chief advantage of this approach is its relative domain independence: facts can be extracted with example sentences rather than requiring the coding of templates.

The system uses a matcher that is fast and general-purpose; it uses "associations" between words (generated from WordNet) to constrain the matches it considers. The matcher alone has no mechanisms guaranteeing the sensibility of the matches it produces; that is a product of both the matcher and the indexer which proposes matches. The "indexer" tries to come up with sensible matches for a new text or a user query. We discussed the matcher's performance using one candidate indexing strategy and provided recall and precision numbers, including an adjustment of the precision for indexing errors.

The preliminary evaluation indicates that an analogical representation is effective for broad-coverage shallow information extraction where expressive patterns are relatively uniform. The success of this sort of extraction is promising for the implementation of more complex understanding systems which uses phrasal matching and indexing as the underpinning for more complicated analogical representations. We are currently designing a system to support just these sorts of representations.

13.11 References

[1] J. Allen. *Natural Language Understanding, second edition.* Benjamin/Cummins, 1995.

[2] W. Andersen, J. Hendler, M. Evett, and B. Kettler. Massively

parallel matching of knowledge structures. In *Parallel Processing for Artificial Intelligence*. Elsevier, 1993.

[3] E. Charniak. *Statistical Language Learning*. MIT Press, 1994.

[4] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied NLP*, 1992.

[5] C. de Marcken. Methods for parallelizing search paths in phrasing. TR 1453, Artificial Intelligence Laboratory, MIT, 1994.

[6] M. Dyer. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. MIT Press, 1983.

[7] T. Evans. A program for the solution of geometric-analogy test questions. In M. Minsky, editor, *Semantic Information Processing*. MIT Press, 1968.

[8] M. Evett, W. Andersen, and J. Hendler. Providing computational effective knowledge representation via massive parallelism. In *Par-*

allel Processing for Artificial Intelligence. Elsevier, 1993.

[9] S. Falhman. *NETL: A Network Representation of Knowledge*. MIT Press, 1979.

[10] B. Falkenhainer, K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

[11] R. French. *The Subtlety of Sameness*. MIT Press, 1995.

[12] D. Gentner. Structure Mapping: A Theoretical Framework for Analogy. *Cognitive Science*, 7(2), 1983.

[13] D. Gentner and K. Forbus. Mac/fac: A model of similarity-based retrieval. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Cognitive Science Society, Lawrence Earlbaum, 1991.

[14] I. Goldstein. FRL: A Frame Representation Language. Memo 333, MIT Artificial Intelligence Laboratory, 1976.

[15] D. Gorfein, editor. *Resolving Semantic Ambiguity.* Springer-Verlag, 1989.

[16] K. Haase. Making clouds from cement: Building abstractions out of concrete examples. In *Proceedings of the US-Japan Worskhop on Integrated Comprehension and Generation in Perceptually Grounded Environments*, 1991. (proceedings in preparation).

[17] J. Hendler. *Integrating marker-passing and problem-solving.* Lawrence Erlbaum, 1988.

[18] G. Hirst. *Semantic Interpretation and the resolution of ambiguity.* Studies in Natural Language Processing. Cambridge University Press, 1987.

[19] G. Hirst. Mixed-depth representations for natural language text. In P. Jacobs, editor, *Text-based Intelligent Systems.* Lawrence Earlbaum, 1992.

[20] J. Hobbs. Representing ambiguity. In *Proceedings of the First West*

Coast Conference on Formal Linguistics. Stanford Linguistics Association, 1982.

[21] R. Hoffman. Monster analogies. *AI Magazine*, 16(3), 1995.

[22] D. Hofstader and the Fluid Analogies Research Group. *Fluid Concepts and Creative Analogies*. Basic Books, 1994.

[23] K. Holyoak and P. Thagard. A computational model of analogical problem solving. In *Similarity and Analogical Reasoning*, pages 242–266. Cambridge University Press, 1989.

[24] K. Holyoak and P. Thagard. *Mental Leaps: Analogy in Creative Thought*. MIT Press, 1995.

[25] P. Jacobs, editor. *Text-based Intelligent Systems*. Lawrence Earlbaum, 1992.

[26] P. Jacobs and L. Rau. SCISOR: Extracting information from online news. *Communcations of the ACM*, 33(11), November 1990.

[27] D. Jones. *Analogical Natural Language Processing.* UCL Press, 1996.

[28] J. Kolodner. *Case Based Reasoning.* Morgan Kaufmann, 1994.

[29] Lehnert, W. et. al. UMASS/HUGHUES: Description of the CIRCUS System used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference.* MUC-5, 1993.

[30] D. B. Lenat and L. Guha. *CYC: Building Large Knowledge-Based Systems.* Digital Press, 1990.

[31] M. McCord, A. Bernth, S. Lappin, and W. Zadrozny. Natural language processing within a slot grammar framework. *International Journal of Artificial Intelligence Tools*, 1(2), 1992.

[32] G. Miller and C. Fellbaum. Semantic networks of english. *Cognition*, (41):197–229, 1991.

[33] Miller, G. et. al. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 1990.

[34] M. Mitchell. *Analogy-making as Perception.* MIT Press, 1993.

[35] MUC-3. *Proceedings of the Third Message Understanding Conference.* Morgan Kaufmann, 1991.

[36] MUC-4. *Proceedings of the Fourth Message Understanding Conference.* Morgan Kaufmann, 1992.

[37] MUC-5. *Proceedings of the Fifth Message Understanding Conference.* Morgan Kaufmann, 1993.

[38] M. Pinkal. *Logic and Lexicon.* Studies in Linguistics and Philosophy. Kluwer Academic, 1995.

[39] M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing.* MIT Press, 1968.

[40] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, January 1986.

[41] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and Application of a Metric on Semantic Nets. *IEEE Transactions on*

Systems, Man, and Cybernetics, 19(1), 1989.

[42] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proccedings of the Eleventh National Conference on Artificial Intelligence.* AAAI-93, 1993.

[43] G. Salton. *A Theory of Indexing.* Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1975.

[44] S. Sato and M. Nagao. Towards Memory-based Translation. In H. Karlgren, editor, *Proceedings of COLING-90*, 1990.

[45] R. Schank. *Dynamic Memory: A Theory of Learning in People and Computers.* Cambridge University Press, 1982.

[46] R. Schank and R. Abelson. *Scripts, Plans, Goals, and Understanding.* Lawrence Earlbaum Associates, 1977.

[47] D. D. Sleator and D. Temperly. Parsing english with a link grammar. Technical Report 91-196, Carnegie Mellon Computer Science

Department, October 1991.

[48] S. Soderland and W. Lehnert. Wrap-up: a trainable discourse module for information extraction. *Journal of AI Research*, 2:131–158, 1994.

[49] M. Stefik. An Examination of a Frame Structured Representation System. In *Proceedings of the Sixth IJCAI.* IJCAI, 1979.

[50] P. Thagard, K. Holyoak, G. Nelson, and D. Gochfeld. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, (46):259–310, 1989.

[51] A. Voutilainen. NPtool, a detector of english noun pharses. In *Proceedings of the Workshop on Very Large Corpora*, 1993.

[52] Voutilainen, A. and Heikkilä, J. and Antilla A. Constraint grammar of english: A performance-oriented introduction. Publication, Department of General Heuristics, University of Helsinki, 1992.

[53] D. Waltz and J. Pollack. Massively parallel parsing. *Cognitive Science*, 9(1), 1976.

[54] P. H. Winston. Learning and reasoning by analogy. *Communcations of the ACM*, 23(12), 1979.

[55] F. W.N. and F. Kučera. *Frequency Analysis of English Usage.* Houghton Mifflin, 1982.

13.12 Notes

1. We were also inspired by the use of multi-scale representations in visual processing to enhance reliability and functionality.

2. Micro-relations are named after **micro-features** [53], this implies the "pre-semantic" character of the relations from which semantic relations are eventually identified.

3. The term "rhizome" comes from Jung's description for the "roots" of human conception. I am grateful to Margaret Benefiel for suggesting this terminology.

4. The use of word roots in indexing is largely equivalent to conventional stem-based indexing.

5. The syntax `meet.verb` is simply a convention indicating the rhizome entry for verb sense of the word "meet" and its morphological variants. In the implementation it is a pointer to a particular frame in the rhizome.

6. ACME stands for "Analogical Constraint Matching Engine"; ARCS

stands for "Analog Retrieval by Constraint Satisfaction".

13.13 Figure Captions

Figure 13-1. The arcs in the figure indicate micro-relations generated by our parser for the sentence "Bill kissed the woman in the library".

Figure 13-2. The associations of the entire sentence are also either rhizome entries (in **bold**) or concepts in the ontology derived from WordNet (in *italics*). We only show the first set of associations in WordNet.

Figure 13-3. Type-unique cognates occur when the elements of two contexts have distinct and unique associations in the global ontology. In this case, distinct associations (solid lines) through a simplified ontology (italics) lead to obvious cognate relations (dotted lines).

Figure 13-4. Triangle cognates occur when the elements of two contexts are associated with a third common context. Associations (solid lines) through distinct elements (in italics) of this third context determine cognate relations (dotted lines) between elements of the two contexts (in bold). For clarity, we have removed the cognate relations reflecting word identity or synonymy.

Figure 13-5. In this situation, cognate matching cannot determine how the human individuals are related to one another, since they're all associated with the prototype *person* (in an ontology simplified for clarity). The grey boxes indicate the matches which the system *is* able to determine by cognate identification alone.

Figure 13-6. When micro-relations are used to reduce the context being considered, cognates relations can be determined. The left hand bounding box represents the reduced context determined by the `subject` micro-relation; the right hand bounding box is likewise derived from the `object` micro-relation. Within these contexts, cognate relations (the grey boxes, as above), are readily determined.