Draft of book chapter later published in: R.Green, C.A. Bean & S.H. Myaeng (Eds.), The semantics of relationships: An interdisciplinary perspective (pp. 161-180). Dordrecht: Kluwer.

Chapter

Identifying Semantic Relations in Text for Information Retrieval and Information Extraction

Christopher Khoo
Division of Information Studies
Nanyang Technological University
Singapore

Sung-Hyon Myaeng
Department of Computer Science
Chungnam National University
Korea

Abstract. Automatic identification of semantic relations in text is a difficult problem, but is important for many applications. It has been used for relation matching in information retrieval to retrieve documents that contain not only the concepts but also the relations between concepts specified in the user's query. It is an integral part of information extraction—extracting from natural language text, facts or pieces of information related to a particular event or topic. Other potential applications are in the construction of relational thesauri (semantic networks of related concepts) and other kinds of knowledge bases, and in natural language processing applications such as machine translation and computer comprehension of text. This chapter examines the main methods used for identifying semantic relations automatically and their application in information retrieval and information extraction.

1. INTRODUCTION

The ability to identify semantic relations in text automatically and accurately is increasingly important in information retrieval and information extraction systems. Information retrieval involves identifying a small subset of documents in a document collection that are likely to contain relevant information in response to the user's query statement. This is performed by matching concepts (typically keywords) in the documents with concepts in the query statement. If the query contains more than one concept, then semantic relationships between the concepts may also be specified in the query. In this case, the relevant documents to be retrieved should contain all the concepts in the query as well as the correct relationships between the concepts. Traditional information retrieval has focused on matching keywords in the text, and ignored the semantic relationships expressed between the concepts. If semantic relationships can be identified accurately in text, this can improve retrieval results for some queries by eliminating the documents containing the required concepts but not the desired relationships between the concepts.

Increasingly, users will not be satisfied with retrieval systems that retrieve whole documents that the user must then read through to identify the portion containing the desired information. Users will want systems that actually extract the information they want (information extraction) and

answer questions (question answering). Retrieval systems will also be expected to help users synthesize information extracted from multiple documents to provide an overview of a subject (information visualization and automated abstraction) or to identify new relationships between facts and synthesize new knowledge (text mining and knowledge discovery). Information extracted from text can also be used to build fact databases and knowledge-bases for various purposes. All these applications require accurate identification of semantic relations in text so that the concepts extracted from a text can be connected with relations to form a coherent whole.

The main approach used to identify semantic relations in text automatically is through some kind of pattern matching. This chapter discusses pattern matching techniques for relation identification and their application in information retrieval and information extraction.

2. AUTOMATIC IDENTIFICATION OF RELATIONS

Identifying semantic relations in text involves looking for certain linguistic patterns in the text that indicate the presence of a particular relation. These patterns may be linear patterns of words and syntactic categories in the text, or graphical patterns in the underlying syntactic structure (parse tree) of the text.

To identify semantic relations in text, pattern-matching is performed to identify the segments of the text or the parts of the sentence parse trees that match with each pattern. If a text segment matches with a pattern, then the text segment is identified to contain the semantic relation associated with the pattern. A pattern typically contains one or more slots or blanks, and the parts of the text that match with the slots in the pattern represent the entities related by the semantic relation.

It is difficult to attain high accuracy in identifying semantic relations in text. The accuracy rate varies wide depending on many factors – the type of semantic relation to be identified, the domain or subject area, the type of text/documents being processed, the amount of training text available, whether knowledge-based inferencing is used, the accuracy of the syntactic pre-processing of the text, etc.

This section discusses the characteristics of linear and graphical patterns, and how they are used to identify semantic relations in text.

2.1 Linear Patterns

The following is an example of a linear pattern that describes one way the cause-effect relation can be expressed in text:

[effect] *is the result of* [cause]

The token with square brackets represent slots to be filled by words/phrases in the text. The slots indicate which part of the sentence represents the *cause* and which part represents the *effect* in the cause-effect relation. The pattern can match with the following sentence:

The fatal accident is the result of drunken driving.

The phrase “the fatal accident” matches with the *effect* slot, and “drunken driving” matches with the *cause* slot.

A linear pattern is thus a sequence of tokens, each token representing one or more of the following:

- *a literal word* in the text, which may or may not be stemmed or converted to a root form
- *a syntactic constituent*, which may be
  - a part-of-speech, e.g. noun, verb and preposition
  - a type-of-phrase, e.g. noun phrase, verb phrase, prepositional phrase and clause
  - a syntactic role, e.g. subject, direct object, indirect object, active-voice verb and passive-voice verb
- *a semantic category*, which may be represented by a concept label from a thesaurus or conceptual hierarchy, or a label representing a meaningful unit, e.g. organization name, person name, date and amount of money
- *a wildcard*, that can match with one or more words in the text
- a pre-defined set of words or syntactic/semantic categories
- a set of sub-patterns
- a slot or blank to be filled in by words in the text.

A token can specify multiple constraints. A token can be a triple <x,y,z>, where

- x is a symbol representing a literal word, a wildcard or a slot
- y is a symbol representing a syntactic constraint
- z is a symbol representing a semantic constraint

A slot in a pattern can thus have a syntactic and/or semantic constraint attached to it.

Table 1 lists some of the patterns used by Khoo, Kornfilt, Oddy and Myaeng (1998) to identify cause-effect relations in text. The patterns listed all involve the phrase *because of*. *[1]* represent the cause slot and *[2]* represent the effect slot. Some slots in the patterns have a syntactic constraint (e.g. noun, noun phrase, etc.) specifying what type of word or phrase may fill the slot.

For each set of patterns, the patterns are tried in the order listed in the set. Once a pattern is found to match a sentence (or some part of a sentence), all the words that match the pattern (except for the words filling the slots) are flagged, and these flagged words are not permitted to match with tokens in any subsequent pattern. So, the order in which patterns are listed in the set is important. As a rule, a more "specific" pattern is listed before a more "general" pattern. A pattern is more specific than another if it contains all the tokens in the other pattern as well as additional tokens not in the other pattern.

Pattern 2 in Table 1 is an example of a negative pattern. If a sentence contains the words “not because”, this will match with pattern 2 and will be flagged so that they will not match with any of the subsequent patterns and be identified as containing a cause-effect relation.

The patterns and the pattern matching process appear simple and straightforward. Each linear pattern is equivalent to a finite-state transition network, and is thus easy to implement. However, a large number of patterns may be needed for a particular application, and so, there must be an

effective method for constructing and managing the set of patterns, and controlling the interaction between related patterns.

Table 1. Examples of linear patterns for identifying the cause-effect relation

No.	Relation	Pattern
1	C	[1] &and because of &this [2] &end_of_clause Example: It was raining heavily and because of this the car failed to brake in time.
2	-	&NOT because Example: It was not because of the heavy rain that the car failed to brake in time.
3	C	it &aux &adv because of [1] that [2] &end_of_clause Example: It was because of the heavy rain that the car failed to brake in time.
4	C	&beginning_of_clause [2] &and_this &aux &adv because of [1] &end_of_clause Example: The car failed to brake in time and this was because of the heavy rain.
5	C	&beginning_of_clause because of [N:1] , [2] Example: John said that because of the heavy rain , the car failed to brake in time.
6	C	&beginning_of_clause [2] because of [1] &end_of_clause Example: The car failed to brake in time because of the heavy rain.

Notes
1. The code in the second column indicates the type of semantic relation associated with the pattern. “C” refers to the cause-effect relation. "-" indicates a negative relation.
2. *[1]* and *[2]* in the patterns represent slots to be filled by the first and second constituent of the relation, the first constituent of the cause-effect relation being the *cause* and the second constituent the *effect*. The type of phrase or word that may fill a slot may also be indicated. The symbol *[N:1]* indicates that the slot for *cause* is to be filled by a noun phrase.
3. The symbol & followed by a label refers to a set of subpatterns (usually a set of synonymous words or phrases). For example, *&aux* refers to auxiliary verbs like *will*, *may*, and *may have been*, and *&beginning_of_clause* refers to subpatterns that indicate the beginning of a clause.

There are also several possible variations in the kinds of patterns used, and how the pattern matching is performed:
1. A pattern may be required to match a *whole* sentence, or just a *portion* of a sentence.
2. A pattern may be allowed to match only *one* portion of a sentence or *multiple* portions of a sentence (i.e. have multiple matches).
3. A pattern may match text *within* a sentence or *across* sentences.
4. A pattern may have a maximum of one, two or more slots
5. A pattern may have only positive constraints or may have negative constraints as well (i.e. may specify that a particular word or syntactic/semantic category not be present in the matching text segment).

6. The set of patterns constructed may be *mutually exclusive*, i.e. if a pattern matches with a text segment, then no other pattern in the set will match the text segment or an overlapping text segment. On the other hand, the patterns may be *overlapping*, i.e. a text segment can match with multiple patterns.
7. If the patterns are overlapping, the pattern matching program can *allow multiple matches* or *single matches* only. With *multiple matches*, a text segment is allowed to match multiple patterns. With *single matches*, a text segment is allowed to match with the first matching pattern only, after which all the matching words are flagged and are not allowed to match with any other pattern. If only single matches are allowed, then the patterns in the set have to be ordered and the patterns are tried in that particular order.
8. The set of patterns may contain positive patterns only, i.e. each pattern is used to identify a particular semantic relation. Or, it may contain negative patterns as well. Negative patterns indicate that if a text segment matches it, then the text segment does not contain the specified semantic relation. When the pattern matching program allows *single matches* only (see item 7), negative patterns can be used to ensure that certain text segments (those that match with the negative patterns) do not match with any of the subsequent positive patterns.
9. If a pattern can contain sub-patterns, then a pattern may be *recursive* (i.e. the sub-pattern may refer to the parent pattern), or non-recursive (i.e. the sub-pattern cannot refer to the parent pattern).

When developing a pattern-matching system for identifying semantic relations, many decisions thus have to be made.

The advantage of using linear patterns as opposed to graphical patterns is that a complete syntactic parse tree of each sentence need not be constructed. Parsing the syntactic structure of sentences is a notoriously difficult problem, and present-day parsers still do not give accurate results for complex sentences. Moreover, only a small segment of a sentence may contain the relation of interest, and parsing a long complex sentence may be largely unnecessary effort.

Linear patterns require only partial syntactic pre-processing of sentences and a small amount of semantic processing. The amount and type of pre-processing required depends, of course, on the nature of the patterns and the kind of syntactic/semantic constraints that may be used in the patterns. One or more of the following kinds of pre-processing may be needed:

1. *Sentence boundary identification*: identifying the beginning and end of sentences
2. *Part-of-speech tagging*: identifying the grammatical category of individual words
3. *Phrase identification and bracketing*: recognizing major phrasal units such as noun phrases, prepositional phrases and clauses
4. *Partial syntactic processing* to identify syntactic roles such as subject and direct object, and active and passive constructions
5. *Semantic tagging*: mapping certain words to the conceptual hierarchy, or identifying the semantic type of a phrase (e.g. company names, place names, currencies, etc.)

2.2 Graphical Patterns

Linear patterns have the advantage of simplicity and the syntactic pre-processing of text required for the pattern matching can be accomplished fairly easily and accurately. The disadvantage is

the large number of patterns that have to be constructed to cover all the superficial variations in sentence constructions.

If a good parser is available, graphical pattern matching that identifies graphical patterns in the syntactic parse trees of sentences can be used. Since different sentence constructions can map to the same parse tree, this can reduce the number of patterns that need to be constructed. Moreover, some difficult syntactic processing such as prepositional attachment (identifying whether a prepositional phrase should be linked to a particular verb or particular noun phrase) may have been resolved by the parser, so that the patterns need not have to handle them.

Graphical patterns are usually in the form of tree structures that are used to match with parts of sentence parse trees. Each node in a graphical pattern can represent a literal word, a semantic category, a slot, a wildcard or a sub-tree. Labeled arcs in the graphical pattern represent syntactic relations between the words.

Khoo et al. (1999 & 2000) used graphical pattern matching to extract cause-effect information from medical abstracts. In their study, each sentence is parsed using Conexor's *Functional Dependency Grammar of English Parser* (http://www.conexor.fi/) to generate a graph representing the syntactic structure of the sentence.

Fig. 1 gives an example of a sentence and the syntactic structure output by Conexor's parser, shown as a graphical diagram as well as in *linear conceptual graph* notation (Sowa, 1984). In this notation, the concept nodes, representing words in the sentence, are in square brackets and the relation nodes, representing syntactic relations, are given in parentheses.

Fig. 1 also provides an example of a graphical pattern that matches with the parse tree of the example sentence, and can be used to extract cause-effect information from the sentence. Within the square brackets, a word in quotation marks represents a stemmed word that must occur in the sentence. A word not in quotation marks (e.g. the word *improve*) refers to a class of synonymous words that can occupy that node. "*" is a wildcard character that can match any term. The roles (or semantic relations) in the cause-effect situation are indicated within the square brackets after the ":" symbol.

2.3 Identifying Case Relations

There are many different types of semantic relations. These can be at different "levels of abstraction." A high level relation can be decomposed into more primitive concepts and relations. The sentence "*John eats an apple*" can be represented in conceptual graph representation (Sowa, 1984) as

[person:"John"] ->(eat)-> [apple]

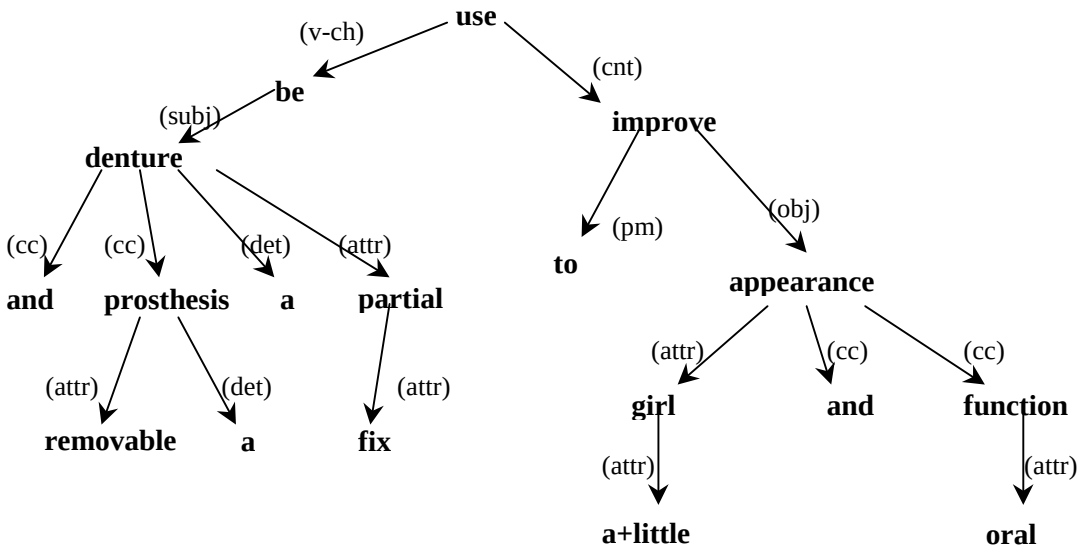
The diagram indicates that John has an "eating" relationship with an apple. The relation *eat* is a high-level relation that can be decomposed into the concept *eat* and the "case relations" *agent* and *patient*:

[person:"John"] <-(agent)<- [eat] ->(patient)-> [apple]

Example sentence

A removable prosthesis and a fixed partial denture are used to improve a little girl's appearance and oral function.

Syntactic structure



Syntactic structure in linear conceptual graph format

[use]-
(v-ch)->[be]->(subj)->[denture]-
(cc)->[prosthesis]-
(det)->[a]
(attr)->[removable] ,
(cc)->[and]
(det)->[a]
(attr)->[partial]->(attr)->[fix] ,
(cnt)->[improve]-
(pm)->[to]
(obj)->[appearance]-
(attr)->[girl]->(attr)->[a+little]
(cc)->[and]
(cc)->[function]->(attr)->[oral] , , , .

Relation node abbreviations

attr:	attributive nominal
cc:	coordinating conjunction
cnt:	contingency
det:	determiner
obj:	object
pm:	preposed marker
subj:	subject
v-ch:	verb chain

The parts of the sentence that match with the graphical pattern below are indicated in bold.

Example pattern

[* : *cause.object*] <-(subj)<- ["be"] <-(v-ch)<- ["use"] ->(cnt)-> [improve : *effect.polarity*] ->(obj)->
[* : *effect.object*]

Information extracted

Cause.object:	denture
Effect.object:	appearance
Effect.polarity:	improve

Fig. 1. Sentence structure and matching pattern in linear conceptual graph format

Case relations are low-level semantic relations that exist between the main verb of a clause and the other constituents of the clause (Fillmore, 1968; Somers, 1987). In the Case Theory, case roles (also called *thematic roles*) are assigned by the verb to the various syntactic constituents of the clause (i.e. the subject, direct object, indirect object, prepositional phrase, etc.). For example, in the sentence,

Mary bought a watch for John.

the case relations between the verb *buy* and the other constituents of the sentence are:

buy –
->(agent)-> [Mary]
->(patient)-> [watch]
->(recipient)-> [John]

Many natural language processing systems work with case relations because case relations correspond closely to the syntactic structure of sentences and are thus easier to extract from text. They are also domain-independent low-level relations that can provide a foundation for any high-level application. Identifying case relations in text serves as a useful intermediate step in the conversion of text to a semantic representation. After the case relations are identified, higher-level relations can be inferred. This section examines how case relations are identified in sentences.

The linguistic patterns used for identifying case relations are called *case frames*. One or more case frames are constructed for each sense or meaning of a verb. A case frame specifies:

1. the number of entities that the verb expects in the sentence
2. the case roles assigned to these entities
3. "selectional restrictions" that specify the semantic category of the entity filling a role
4. the syntactic position of each entity in the sentence
5. whether each role is obligatory (i.e. must be filled) or optional.

Table 2. Example case frames

Frame no.	Verb	Sense no.	Case role	Syntactic category	Semantic category	Obligatory / Optional
1	Buy	1	Agent	Subject	Human	Obligatory
			Patient	Object	Concrete	Obligatory
			Recipient	Prepositional phrase "for"	Human	Obligatory
2	Decline (e.g. "The stock prices declined")	1	Patient	Subject	-	Obligatory
3	Decline (e.g. "She declined the offer")	2	Agent	Subject	Human	Obligatory
			Patient	Object	-	Optional
4	Decline (e.g. "She declined to go.")	3	Agent	Subject	Human	Obligatory
			Activity	Infinitive	-	Obligatory

Table 2 gives examples of case frames for the verbs *buy* and *decline*. The first case frame (for "buy") indicates that the subject of the verb has an *agent* role, and the entity filling this role must belong to the semantic class of human entities. The entity assigned the patient role must be a concrete object. The recipient role is assigned to the object of the preposition "for". All the roles are obligatory, indicating that if all the roles are filled and the entities filling the roles satisfy the syntactic and semantic constraints, then the case frame is instantiated and there is a match. We can then infer that the semantic relations between the main verb and the various constituents of the clause are the case relations specified in the case frame.

If there is no match, then the system has to find another case frame to match the clause. If more than one case frame match a clause, then the system has to select the frame that best matches it. Myaeng, Khoo & Li (1994) used the following rules to select the best frame:

- The frame with the smallest number of obligatory rules left unfilled
- The frame with the highest number of roles filled (i.e. the case frame which accounts for the highest number of constituents in the sentence)
- The case frame for the most commonly used sense of the verb.

Whereas a complete case frame specifies all the case roles assigned by the verb, researchers have worked with linear patterns that can be characterized as partial case frames that seek to identify just one case relation at a time. The following example patterns are derived from Cowie & Lehnert (1996):

- <"robbed", active-voice verb> <slot:robbery.victim, noun phrase>
  The noun phrase following "robbed" (active voice) refers to the robbery victim.
- <slot:kidnapping.victim, noun phrase> <"disappeared", active-voice verb>
  The noun phrase preceding "disappeared" (active voice) refers to the victim of a kidnapping.
- <"travelling", active-voice verb> <"in"> <slot:attack.target, noun phrase>
  The noun phrase following the preposition "in" after "travelling" (active voice) is the target of an attack (e.g. "The group was travelling in a four-wheel-drive vehicle.")
- <slot:attack.instrument, subject> <"hurled", passive-voice verb>
  The noun phrase preceding "hurled" (passive voice) is the instrument of an attack (e.g. "Dynamite sticks were hurled from a car.").

Each of the patterns has only one slot. The generic case relation labels (*agent*, *patient*, etc.) had been replaced in the example patterns with higher-level domain-specific role labels that were more meaningful in the context of the application—extracting information on terrorist acts. The *patient* role for the verb *rob* was re-labelled *robbery.victim*, and the patient role for the verb *disappeared* was re-labelled *kidnapping.victim*.

3. PATTERN CONSTRUCTION

3.1 Approaches to Pattern Construction

Construction of patterns can be done manually or automatically by analyzing sample relevant text and the associated answer key indicating the semantic relation present in the text and the

concepts in the text connected by the relation. The answer keys are typically constructed by human analysts that have to be trained for the task.

Pattern construction thus entails constructing patterns that would extract the same information from the text as the analysts did. The patterns constructed have to be general enough to extract correct information from unseen text segments that are "similar" to but not exactly the same as those in the training text. At the same time, the patterns should not be too general as to extract information from non-relevant text fragments or extract incorrect information from relevant text fragments. The set of patterns should be parsimonious: there should be enough patterns to give good results but not more patterns than necessary, as this will increase processing time.

Two approaches can be used in the pattern construction:
1. *general-to-specific approach*: general patterns are first constructed which are gradually specialized to reduce errors
2. *specific-to-general approach*: specific patterns are first constructed which are gradually generalized to cover more situations.

The *general-to-specific approach* constructs a general pattern (pattern with the least number of constraints) from the training text. More and more constraints are added to this general pattern as negative examples are encountered. Negative examples refer to text segments that would match with the pattern, but yield incorrect information. Constraints are added to the general pattern so that it will either extract the correct information from a negative example or would not match the example at all. At the same time, care is taken to ensure that the pattern will continue to extract the correct information from previously encountered positive examples.

To specialize a general pattern, the pattern can be modified in the following ways:
1. Add a syntactic or semantic constraint to one or more tokens in the pattern
2. Replace a semantic constraint with a more specific semantic category (e.g. change "animate" to "human")
3. Add one or more tokens to the pattern.

The general-to-specific approach is thus an optimistic approach that favors recall, i.e. it tries not to miss any relevant information at the cost of precision. It may extract spurious information or extract more information than is correct.

The *specific-to-general approach* is a conservative approach in which a pattern is constructed that will match only text fragments that very closely resembles the training text. As new example text fragments are found that indicate that the pattern is not general enough, then the constraints are gradually relaxed or removed to accommodate the new examples. These new examples can be characterized as positive examples. They do not match with the current set of patterns, which have to be modified to "cover" them and extract correction information from them. This approach favors precision at the cost of recall.

The generalize a pattern to cover more example text segments, the pattern can be modified in the following ways:
1. Remove a syntactic or semantic constraint from a token

2. Convert a literal token (i.e. a token that has to match an exact word in the text) to a syntactic or semantic category.
3. Replace a semantic constraint with a more general semantic category (e.g. change “human” to “animate”)
4. Add an alternative syntactic or semantic constraint, i.e. replace a single constraint with a disjunction of two or more alternative constraints so that only one of the constraints need to be met. For example, a token could specify a noun phrase OR a prepositional phrase.

Another way to generalize a set of patterns, is to construct a new pattern to cover the new example text segment.

In practice, some combination of these two approaches are used in order to balance recall and precision. Based on one or more similar text segments and the associated answer keys, one or more patterns are constructed that represent an estimate of the optimal level of generality and specificity. As positive and negative example texts are encountered, the patterns are generalized and specialized to improve coverage while keeping the error rate low. If the pattern construction is performed by human analysts, the analysts will use their linguistic knowledge and judgment to construct patterns at a good level of generality and specificity, perhaps with the help of some guidelines for consistency.

However, some researchers have successfully used automatic or machine-aided means of pattern construction. Automatic pattern construction probably requires a larger set of training examples with answer keys created by human analysts. Creating answer keys is probably faster and requires less training than constructing a good set of patterns. Moreover, recall and precision probabilities can be used to guide the automatic pattern construction to get optimal results.

To perform automatic pattern construction, the system needs well-defined heuristics for constructing the initial patterns, and for generalizing and specializing the patterns based on positive and negative examples. We have listed above several ways of generalizing and specializing a set of patterns. The system will need heuristics for selecting which generalization/specialization method to use in which situation and the order in which the methods are tried. Typically, a variation of the inductive learning algorithm described by Michalski (1983) and Mitchell (1982) is used. In the rest of the section, we examine the heuristics used by three systems, AutoSlog, PALKA and CRYSTAL, for automatic pattern construction.

3.2 Automatic Pattern Construction in Three Systems

The AutoSlog system (Riloff, 1993 & 1996) uses partial case frames as linear patterns. Each pattern has only one slot, and usually includes a verb and a noun phrase (a subject or direct object). A set of pattern templates define the linear patterns that the system will construct. Each pattern is thus an instantiation of a pattern template. Table 3 lists the pattern templates used in AutoSlog and an example pattern for each template.

Before pattern construction, the training corpus has to be preprocessed by identifying clause boundaries and the major syntactic constituents—subject, verb, direct object, noun phrases and

prepositional phrases. Relevant text segments that contain the semantic relations of interest are identified, and answer keys are constructed to indicate which noun phrase should be extracted and the semantic role it has. If the domain of interest is terrorist activities, the semantic roles would include *perpetrators*, *targets*, *victims*, etc.

During pattern construction, pattern matching is used to match the pattern templates with the training text segments. If a pattern template matches with a relevant text segment, then a pattern is constructed by replacing the tokens in the template with the words in text. If a token in the template indicates a slot, this token is allowed to match a noun phrase in the text only if the noun phrase appears in the answer key (i.e. a human analyst has indicated that this is the information to be extracted). A slot token is placed in the pattern being constructed, and the semantic role for the slot is taken from the answer key. The constructed pattern is thus a specialization of the pattern template. Finally, a human analyst manually inspects each pattern and decides which ones should be accepted or rejected.

Table 3. Pattern templates and example of instantiated patterns in AutoSlog

Pattern Template	Example of Pattern Constructed
<subj:slot> <passive-verb>	[victim] was murdered
<subj:slot > <active-verb>	[perpetrator] bombed
<subj:slot > <verb> <infinitive-phrase>	[perpetrator] attempted to kill
<subj:slot > <auxiliary-verb> <noun>	[victim] was victim
<passive-verb> <direct-obj:slot>	killed [victim]
<active-verb> <direct-obj:slot>	bombed [target]
<infinitive-phrase> <direct-obj:slot>	to kill [victim]
<verb> <infinitive-phrase> <direct-obj:slot>	tried to attack [target]
<gerund> <direct-obj:slot>	killing [victim]
<noun> <auxiliary-verb> <direct-obj:slot>	fatality was [victim]
<noun> <preposition> <noun-phrase:slot>	bomb against [target]
<active-verb> <preposition> <noun-phrase:slot>	killed with [instrument]
<passive-verb> <preposition> <noun-phrase:slot>	was aimed at [target]

AutoSlog does not adjust the patterns by generalizing or specializing them once they are constructed. We now look at two systems, PALKA and CRYSTAL, that do adjust the patterns after the initial construction.

In the PALKA system (Kim, 1996; Kim & Moldovan, 1995), the patterns involve the whole clause. Sentences in the training text are first converted to simple clauses. The clauses containing a semantic relation of interest are processed one at a time. If the set of patterns already constructed do not match a clause, then a new pattern is constructed for the clause. This initial pattern covers the main verb, the subject, the object and the words to be extracted (i.e. the slot). Each of these constituents in the clause is represented by a token in the pattern. Each token is assigned a semantic category from a conceptual hierarchy.

Generalizations and specializations of the patterns are applied only to the semantic constraints. When two similar patterns are generated, their semantic constraints are generalized by locating a

broader concept or ancestor in the conceptual hierarchy that is common to both semantic categories. If it is not possible to find a common broader concept, a disjunction of concepts (i.e. a set of alternative concepts) is specified as the semantic constraint. When a negative example text is found indicating that a pattern is over-generalized, the semantic constraint is specialized by finding a disjunction of concepts that covers all the positive examples but excludes the negative example.

The CRYSTAL system (Soderland et al., 1996) uses a similar approach, but is more complex. The text is also pre-processed to identify the major syntactic constituents, subject, verb phrase, direct & indirect object and prepositional phrases. For each word in the text, its semantic class in a conceptual hierarchy is also identified.

Initially, a very specific pattern is constructed for every sentence in the training text. The sentences are not simplified into simple clauses. The constraints in the initial patterns are gradually relaxed to increase their coverage and to merge similar patterns. CRYSTAL identifies possible generalizations by locating pairs of highly similar patterns. This similarity is measured by counting the number of relaxations required to unify the two patterns. A new pattern is created with constraints relaxed just enough to unify the two patterns—dropping constraints that the two do not share and finding a common ancestor of their semantic constraints. The new pattern is tested against the training corpus to make sure it does not extract information not specified in the answer keys. If the new pattern is valid, all the patterns covered by the new pattern are deleted. This generalization proceeds until a pattern that exceeds a specified error threshold is generated.

The next section surveys how the automatic identification of relations is used in information retrieval and information extraction.

4. APPLICATIONS

4.1 Relation Matching in Information Retrieval

As mentioned earlier, information retrieval is typically performed by matching concepts (usually keywords) in the query with concepts in the documents. A document will be retrieved if it contains the query keywords even if the keywords do not have the desired relation between them as expressed in the query. In relation matching, both the concepts and the *relations* between the concepts expressed in the query are matched with concepts and relations in documents. It appears plausible that relation matching should improve information retrieval effectiveness because it provides additional evidence for predicting the document’s relevance.

Early researchers in information retrieval have attempted to perform relation matching using manually identified relations. In some indexing and abstracting databases, documents are indexed using manually assigned indexing terms that are "precoordinated", i.e. some human indexer has indicated that there is a relationship between the concepts in the document. The type of relation is usually not specified in precoordinated indexing but is implied by the context. This is the case with faceted classification schemes like Precis (Austin, 1984) and Ranganathan's Colon classification

(Kishore, 1986; Ranganathan, 1965). Precoordinate indexing allows the user to specify that there is some kind of relation between two terms when searching the database.

Farradane (1967) advocated the use of explicitly specified relations in the indexing system. He pointed out that implied relations in precoordinate indexing are unambiguous only in a narrow domain. In a heterogenous database covering a wide subject area, the relation between the precoordinated terms may be ambiguous. Two indexing systems that make explicit use of relations are Farradane's (1950, 1952 and 1967) relational classification system and the SYNTOL model (Gardin, 1965; Levy, 1967). It is not known whether the use of explicit relations in indexing really improves retrieval effectiveness compared with using individual index terms or using implicit relations.

Researchers have attempted to identify syntactic and semantic relations in text automatically for improving information retrieval effectiveness. Some studies have found a small improvement when syntactic relations are taken into account in the retrieval process (Croft, 1986; Croft, Turtle & Lewis, 1991; Dillon & Gray, 1983; Smeaton & van Rijsbergen, 1988). *Syntactic relations* refer to relations that are derived from the syntactic structure of the sentence. However, the retrieval performance from syntactic relation matching appears to be no better than and often worse than the performance obtainable using index phrases generated using statistical methods based on word proximity or co-occurrence in text, such as those described in Salton, Yang and Yu (1975) and Fagan (1989).

Semantic relations, if identified accurately, can yield better retrieval results than syntactic relations. This is because a semantic relation can be expressed using different syntactic structures. A semantic relation is partly but not entirely determined by the syntactic structure of the sentence. Syntactic relation matching can fail to find a match between similar semantic relations if the relations are expressed using different syntactic structures.

Most of the information retrieval systems that automatically identify semantic relations are information extraction systems, described in the next section. However, some researchers have studied particular types of semantic relations. Lu (1990) investigated the use of *case relation* matching using a small test database of abstracts. Using a tree-matching method for matching relations, he obtained worse results than from vector-based keyword matching.

Liu (1997) has investigated *partial relation matching*. Instead of trying to match the whole concept-relation-concept triple (i.e. both concepts as well as the relation between them), he sought to match individual concepts together with the semantic role that the concept has in the sentence. In other words, instead of trying to find matches for "term1 ->(relation)-> term2", his system sought to find matches for "term1 ->(relation)" and "(relation)-> term2" separately. Liu focused on case relations, and was able to obtain positive results only for long queries (abstracts that were used as queries).

Khoo, Myaeng & Oddy (2000) carried out an in-depth study of just one relation—the cause-effect relation. Causal relation matching did not yield better retrieval results than word proximity matching. However, as with the study by Liu (1997), partial relation matching where one concept in the relation is replaced with a wildcard was found to be helpful. Thus, if a query contains the

relation “smoking causes cancer”, then better retrieval results are likely to be obtained if we look for “smoking ->(cause)-> *” and “* ->(cause)-> cancer” (where “*” is a wildcard), rather than “smoking ->(cause)-> cancer”. Substituting a wildcard for one constituent of the relation is effectively the same as assigning semantic roles to terms, as used by Liu (1997) and Marega and Pazienza (1994).

This suggests that relation matching in information retrieval may be useful for queries in which one concept in the relation is not specified, but is expected to be supplied by the relevant document, as in this example query:

I want documents that describe the consequences of the Gulf War.

This query can be represented as follows:

Gulf War ->(cause)-> *

4.2 Relations in Information Extraction

Whereas information retrieval seeks to identify documents that are likely to contain relevant information, information extraction goes further to identify the passage(s) in the document containing the desired information, extract the pieces of information and relate them to one another by filling a structured template or a database record. The goal of information extraction is to find and relate relevant information while ignoring extraneous and non-relevant information (Cardie, 1997; Cowie & Lehnert, 1996; Gaizauskas & Wilks, 1998).

An information retrieval system processes ad hoc queries, i.e. the user enters any query into the system and expects to get the documents almost instantly. On the other hand, an information extraction system requires extensive training on the topic of interest before it can extract relevant information on that topic. An extensive knowledge-base has to be custom-developed for the topic. The knowledge-base will include a set of linguistic patterns for extracting relational information (as described in Section 2 and 3), and possibly a conceptual hierarchy for the domain. Topics used for information extraction topics have been characterized as “frozen queries” (Gaizauskas & Wilks, 1998). They persist over a period of time and must be of interest to a community of users, since substantial effort is required to train the system on the topic.

Research in information extraction has been influenced tremendously by the series of Message Understanding Conferences (MUC), organized by the U.S. Advanced Research Projects Agency (ARPA). Participants of the conferences develop systems to perform common information extraction tasks, defined by the conference organizers.

For each task, a template is specified that indicates the slots to be filled in. The type of information to be extracted to fill each slot is specified, and the set of slots defines the various entities, aspects and roles relevant to the topic of interest. An example template used in the MUC-3 Conference for terrorist activities is given in Table 4. More recent MUC conferences have used more complex object-oriented templates in which a slot-fill can be the exact words extracted from the text (“string fill”), one of a given set of alternatives (“set fill”), words extracted from the text and transformed into a canonical form, e.g. dates and monetary amounts (“normalized entries”), and pointers to other objects (“references”).

Table 4. Example of a filled template for terrorist acts

Text

Bogota, 30 Aug 89 (Inravision Television Cadena 2) – Last night's terrorist target was the Antioquia Liqueur Plant. … The watchmen on duty reported that at 2030 they saw a man and a woman leaving a small suitcase near the fence that surrounds the plant. The watchmen exchanged fire with the terrorists who fled leaving behind the explosive material that also included dynamite and grenade rocket launchers. Metropolitan Police personnel specializing in explosives defused the rockets. Some 100 people were working inside the plant.

Filled template

MESSAGE ID:	TST1-MUC3-0004
TEMPLATE ID:	1
DATE OF INCIDENT:	29 Aug 89
TYPE OF INCIDENT:	Attempted bombing
CATEGORY OF INCIDENT:	Terrorist act
PERPETRATOR: ID OF INDIV(S):	"man", "woman"
PHYSICAL TARGET: ID(S):	"Antioquia Liqueur Plant"
PHYSICAL TARGET: TOTAL NUM:	1
PHYSICAL TARGET: TYPE(S):	Commercial
HUMAN TARGET(S): ID(S):	"people"
HUMAN TARGET:TOTAL NUM:	Plural
HUMAN TARGET: TYPE(S)	Civilian
LOCATION OF INCIDENT:	Colombia: Antioquia
EFFECT ON PHYSICAL TARGET(S):	No damage: "Antioquia Liqueur Plant"
EFFECT ON HUMAN TARGET(S):	No injury or death: "people"

(Source: MUC-3, 1991, Appendix D and E)

The process of information extraction can be seen as a form of automatic identification of relations between entities, events and situations. Whereas in Section 2 and 3, we focused on simple relations between two entities/events/situations or a set of case relations involving a verb and other constituents of a clause, information extraction seeks to identify a web of relationships relating to a particular type of event, situation or topic. These web of relations are encoded in the structure of the template, showing how the various pieces of information are related to one another.

Although information extraction seeks to identify many kinds of relations that are relevant to the topic of interest, the approach used for identifying relations is similar to that described earlier in the Chapter. Typically, each *concept->relation->concept* triple is identified one at a time using pattern matching. The *concept-relation-concept* triples are merged or linked together into a bigger structure using discourse processing, co-reference resolution, knowledge-based inferencing or assumed to be related simply because they occur in close proximity in the document.

Information that has been extracted can be used for several purposes: for populating a database of facts about entities (e.g. companies) or events, for automatic summarization, for information

mining, and for acquiring knowledge to use in a knowledge-based system. Information extraction systems have been developed for a wide range of tasks, including:

- extracting information about international terrorist events from newswire stories (MUC-3, 1991)
- extracting information about corporate mergers, acquisitions and joint ventures from newspaper text (Rau, 1987; Rau, Jacobs & Zernik, 1989; MUC-5, 1994)
- extracting information about developments in semiconductor fabrication techniques (MUC-5, 1993)
- extracting the details of patent claims from patent documents to store in a relational database (Nishida & Takamatsu, 1982)
- extracting information from X-ray reports for the purpose of indexing and retrieving the reports and pictures (Berrut, 1990)
- extracting information from patient medical records and discharge summaries (Sager, 1981; Sager, et al., 1994; Soderland et al., 1995)
- extracting information from life insurance applications (Glasgow et al., 1998).

How accurate are information extraction systems? This depends on the difficulty and complexity of the information extraction task. For fairly complex information extraction tasks, one can expect an accuracy of less than 60% measured using an F-measure (Van Rijsbergen, 1979) that weights recall and precision equally (MUC-6, 1995; MUC-7, 2000). The precision is usually below 70% and the recall usually below 60% for the best systems. Cowie and Lehnert (1996) suggest that 90% precision will be necessary for information extraction systems to satisfy users.

5. CONCLUSION

This chapter has examined how semantic relations in text are identified and extracted automatically using pattern matching with a set of linguistic patterns. We have analyzed the different types of patterns and pattern matching that can be used, as well as the different ways of constructing the patterns automatically using inductive learning. Application to information retrieval and information extraction was also surveyed.

Information retrieval systems can process adhoc queries in real time, but perform retrieval only at the document level. Information extraction systems can extract specific information but only after extensive training. The big challenge is to combine information retrieval and information extraction technologies in a way that takes advantage of their individual strengths so that information extraction and question answering can be performed in real time for adhoc queries.

Automatic extraction of semantic relations from text is an exciting and increasingly important area for research and development because of the tremendous amount of textual information and textual databases accessible on the World Wide Web. Information extraction for specific topics and applications has been shown to be feasible in the series of MUC conferences. Information extraction can be applied to the Web for building knowledge bases for various kinds of applications.

In particular, the World Wide Web presents tremendous opportunities for information mining and knowledge discovery. Techniques for identifying semantic relationships and extracting information can be used for automatic summarization of Web documents. Information extracted from Web documents can be chained or connected to give a conceptual map of the information available on a particular topic. The conceptual map can show how information in one document is related to information in another document, and how the documents are related in content. This conceptual map can support creativity by suggesting hypotheses for investigation and indicating gaps in knowledge.

6. REFERENCES

Austin, D. (1984). *PRECIS: A manual of concept analysis and subject indexing (*2nd ed.). London: British Library, Bibliographic Services Division.

Berrut, C. (1990). Indexing medical reports: The RIME approach. *Information Processing & Management, 26*(1), 93-109.

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine, 18*(4), 65-79.

Cowie, J., & Lehnert, W. (1996). Information extraction. *Communications of the ACM, 39*(1), 80-91.

Croft, W.B. (1986). Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science, 37*(2), 71-77.

Croft, W.B., Turtle, H.R., & Lewis, D. (1991). The use of phrases and structured queries in information retrieval. In *SIGIR '91: Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval* (pp. 32-45). New York: ACM Press.

Dillon, M., & Gray, A.S. (1983). FASIT: A fully automatic syntactically based indexing system. *Journal of the American Society for Information Science, 34*(2), 99-108.

Fagan, J.L. (1989). The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science, 40*(2), 115-132

Farradane, J.E.L. (1950). A scientific theory of classification and indexing and its practical applications. *Journal of Documentation, 6*(2), 83-99.

Farradane, J.E.L. (1952). A scientific theory of classification and indexing: further considerations. *Journal of Documentation, 8*(2), 73-92.

Farradane, J.E.L. (1967). Concept organization for information retrieval. *Information Storage and Retrieval, 3*(4), 297-314.

Fillmore, C.J. (1968). The case for case. In E. Bach & R. T. Harms (Eds.), *Universals in Linguistic Theory* (pp.1-88). New York: Holt, Rinehart and Winston.

Gaizauskas, R., & Wilks, Y. (1998). Information extraction beyond document retrieval. *Journal of Documentation, 54*(1), 70-105.

Gardin, J.-C. (1965). *SYNTOL.* New Brunswick, NJ: Graduate School of Library Service, Rutgers University.

Glasgow, B., Mandell, A., Binney, D., Ghemri, L., & Fisher, D. (1998). MITA: an information-extraction approach to the analysis of free-form text in life insurance applications. *AI Magazine, 19*(1), 59-71

Khoo, C., Chan, S., Niu, Y., & Ang, A. (1999). A method for extracting causal knowledge from textual databases. *Singapore Journal of Library & Information Management, 28*, 48-63.

Khoo, C., Chan, S., & Niu, Y. (2000). Extracting causal knowledge from a medical database using graphical patterns. In *ACL-2000: 38th Annual Meeting of the Association for Computational Linguistic* (pp. 336-343). New Brunswick, NJ: Association for Computational Linguistics.

Khoo, C., Kornfilt, J., Oddy, R., & Myaeng, S.H. (1998). Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary & Linguistic Computing, 13* (4), 177-186.

Khoo, C., Myaeng, S.H., & Oddy, R. (2000). Using cause-effect relations in text to improve information retrieval precision. *Information Processing & Management*, 37(1), 119-145.

Kim, J.-T. (1996). Automatic phrasal pattern acquisition for information extraction from natural language texts. *Journal of KISS (B), Software and Applications, 23*(1), 95-105.

Kim, J.-T., & Moldovan, D.I. (1995). Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering, 7*(5), 713-724.

Kishore, J. (1986). *Colon classification: Enumerated & expanded schedules along with theoretical formulations*. New Delhi: Ess Publications.

Levy, F. (1967). On the relative nature of relational factors in classifications. *Information Storage & Retrieval, 3*(4), 315-329.

Liddy, E.D., & Myaeng, S.H. (1993). DR-LINK's linguistic-conceptual approach to document detection. In *The First Text REtrieval Conference (TREC-1)* (NIST Special Publication 500-207, pp. 1-20). Gaithersburg, MD: National Institute of Standards and Technology.

Liu, G.Z. (1997). Semantic vector space model: Implementation and evaluation. *Journal of the American Society for Information Science, 48*(5), 395-417.

Lu, Xin. (1990). An application of case relations to document retrieval (Doctoral dissertation, University of Western Ontario, 1990). *Dissertation Abstracts International, 52-10,* 3464A.

Marega, R., & Pazienza, M.T. (1994). CoDHIR: An information retrieval system based on semantic document representation. *Journal of Information Science, 20*(6), 399-412.

Michalski, R. (1983). A theory and methodology of inductive learning. *Artificial Intelligence, 20*, 111-161.

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence, 18*, 203-226.

MUC-3. (1991). *Third Message Understanding Conference (MUC-3).* San Mateo, CA: Morgan Kaufmann.

MUC-4. (1992). *Fourth Message Understanding Conference (MUC-3).* San Mateo, CA: Morgan Kaufmann.

MUC-5. (1993*). Fifth Message Understanding Conference (MUC-5).* San Francisco: Morgan Kaufmann.

MUC-6. (1995*). Sixth Message Understanding Conference (MUC-6).* San Francisco: Morgan Kaufmann.

MUC-7. (2000). *Message Understanding Conference proceedings (MUC-7)* [Online]. Available: http://www.muc.saic.com/proceedings/muc_7_toc.html.

Myaeng, S.H., & Liddy, E.D. (1993). Information retrieval with semantic representation of texts. In *Proceedings of the 2nd Annual Symposium on Document Analysis and Information Retrieval* (pp. 201-215).

Myaeng, S.H., Khoo, C., & Li, M.. (1994). Linguistic processing of text for a large-scale conceptual information retrieval system. In *Conceptual Structures: Current Practices: Second International Conference on Conceptual Structures, ICCS '94* (pp. 69-83). Berlin: Springer-Verlag.

Nishida, F., & Takamatsu, S. (1982). Structured-information extraction from patent-claim sentences. *Information Processing & Management, 18*(1), 1-13.

Ranganathan, S.R. (1965). *The Colon Classification*. New Brunswick, N.J.: Graduate School of Library Service, Rutgers University.

Rau, L. (1987). Knowledge organization and access in a conceptual information system. *Information Processing & Management, 23*(4), 269-283.

Rau, L.F., Jacobs, P.S., & Zernik, U. (1989). Information extraction and text summarization using linguistic knowledge acquisition. *Information Processing & Management, 25*(4), 419-428.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 811-816). Menlo Park, CA: AAAI Press.

Riloff, E. (1996). An empirical study of automated dictionary construction for information extraction in three domains. *Artifical Intelligence, 85*(1-2), 101-134

Sager, N. (1981). *Natural language information processing: A computer grammar of English and its applications*. Reading, MA: Addison-Wesley.

Sager, N, Lyman, M., Tick, L.J., Ngo Thanh Nhan, Bucknall, C.E. (1994). Natural language processing of asthma discharge summaries for the monitoring of patient care. In *Proceedings of Seventeenth Annual Symposium on Computer Applications in Medical Care* (pp. 265-268). New York: McGraw-Hill .

Salton, G., Yang, C.S., & Yu, C.T. (1975). A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science, 26*(1), 33-44.

Soderland, S, Aronow, D., Fisher, D., Aseltine, J., & Lehnert, W. (1995). *Machine learning of text-analysis rules for clinical records* (Technical Report, TE-39). Amherst, MA: University of Massachusetts, Dept. of Computer Science.

Soderland, S., Fisher, D., Aseltine, J. & Lehnert, W. (1996). Issues in inductive learning of domain-specific text extraction rules. In S. Wermter, E. Riloff, & G. Scheler (Eds.), *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing* (pp. 290-301). Berlin: Springer-Verlag.

Smeaton, A.F., & van Rijsbergen, C.J. (1988). Experiments on incorporating syntactic processing of user queries into a document retrieval strategy. In Y. Chiaramella (Ed.), *11th International Conference on Research & Development in Information Retrieval* (pp. 31-51). New York: ACM.

Somers, H.L. (1987). *Valency and case in computational linguistics*. Edinburgh : Edinburgh University Press.

Sowa, J.F. (1984). *Conceptual structures: Information processing in mind and machine*. Reading, MA: Addison-Wesley.

Van Rijsbergen, C.J. (1979). *Information retrieval*. London: Butterworths.