34. Searching treebanks and other structured corpora

1. Introduction
2. Elementary topology
3. Representation, visualisation and matching
4. Case interaction
5. Advanced topology
6. Conclusions
7. Acknowledgements
8. Literature

1. Introduction

A corpus is a collection of written or spoken text compiled for the purposes of linguistic description and analysis. In a parsed corpus, each sentence is given a structured grammatical analysis in the form of a tree. For linguistic purposes it is necessary to augment parsing with manual correction. Automatic parsing of unrestricted text tends to produce incomplete analyses (Briscoe 1996), although these may prove useful in computational applications.

In this book we use the term *treebanks* to refer to parsed corpora whose sentences have been constructed or verified by a linguist. These corpora are (a) much more limited in scale than automatically tagged megacorpora such as the BNC (Aston/Burnard 1998) − a million words is typical, (b) require significant effort to construct and (c) have a wide range of uses from simple exemplification to the evaluation of linguistic theories. As these datasets are collected and annotated, questions arise as to how best to exploit and explore them.

Some applications, such as identifying cases for teaching purposes, simply require the extraction of suitable examples. Others, including general linguistic research and computational generalisation, have rather more complex requirements. These can be considered in two distinct stages:

(i) **Repurposing** the data by focusing on concepts central to a particular research programme or application goal (including designing experiments and extracting a relevant dataset). A query defines a set of results that can then be further evaluated.
(ii) **Evaluating** this data against linguistic hypotheses (Wallis/Nelson 2001) or otherwise generalising from the dataset (articles 42 and 43).

This article is concerned with searching large forests of annotated data. In so doing, we distinguish between an annotation scheme *per se* and the general approach one takes to specifying a query. The critical question is, *by what procedure, and employing which representation, should researchers comb this forest of utterances for linguistic knowledge?*

This paper is organised as follows. Section 2 discusses the *topology* of parsing schemes. By topology we mean the set of structural constraints which define permissible trees and queries. Whereas grammatical schemes differ widely (cf articles 13, 28), structural constraints applied to them vary less. The two most common topologies are *con-*

stituent grammars, including phrase structure grammars (e. g., Marcus et al. 1993; Nelson/Wallis/Aarts 2002), and *dependency grammars*, including constraint grammars (e. g., Karlsson et al. 1995).

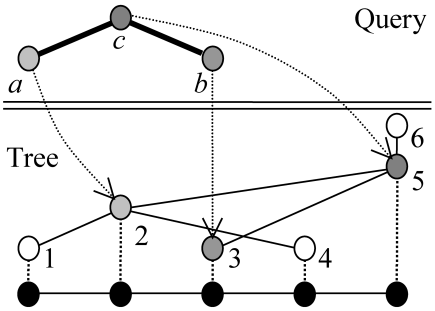


Fig. 34.1: Matching a query to a tree so that <*a*, *b*, *c*> = <2, 3, 5>

Section 3 discusses the relative merits of employing formal logic and diagrammatic models for the purposes of composing a structured query, and the visualisation of resulting cases in a corpus. We distinguish between the process of *matching* a query to a tree in the corpus (Figure 34.1) and any subsequent processes (depending on the application) that may organise and *evaluate* these matching cases. Matching a query against a tree means applying a *proof procedure* to identify configurations of nodes and words (such as <2, 3, 5>) that correspond to a query (<*a*, *b*, *c*>).

Reliably retrieving examples of phenomena from a corpus is not merely a question of applying a single query and yielding a single result. Section 4 considers the problem of retrieving multiple results that may interact with, or even overlap, one another.

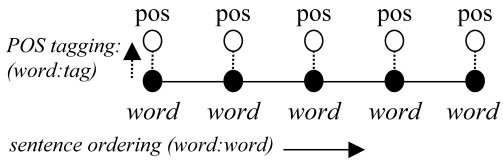
Finally, section 5 considers how extending the annotation of corpora impacts on problems of search.

2. Elementary topology

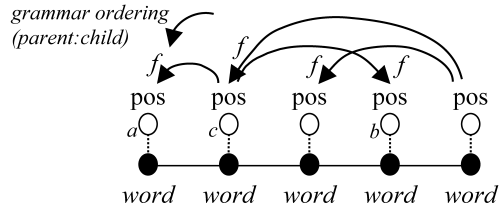
A parsed corpus is segmented into plausible sentences annotated in the form of a tree. In this section we consider corpora with the three distinct topologies summarised in Figure 34.2: (a) part of speech (POS) tagging, (b) dependency or constraint grammar and (c) constituent phrase structure grammar. These three representations broadly cover the gamut of parsed corpora. Section 5 reviews some more complex structural issues.

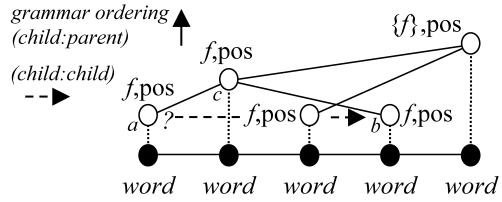
2.1. A POS-tagged corpus

In Figure 34.2(a), a string of lexical items is connected in sequence. Each word is given a part of speech tag, marked as 'pos' nodes. Typically, this tag will contain a word class category (noun, verb etc.) and further subcategories (plural, past, etc). Different tagsets denote subcategories differently (cf. article 23). CLAWS derivatives such as C5 (McEnery/Wilson 2001) label a common singular noun 'NN1', while ICE (Nelson/Wallis/Aarts

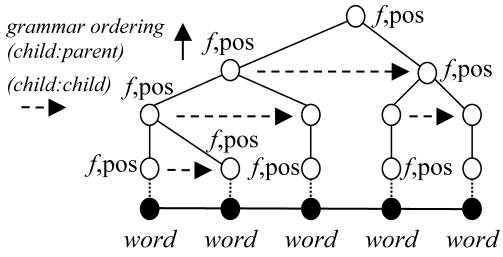


(a) simple part-of-speech tagged sentence





(b) depency or constraint grammar: alternative visualisations, identical topology



(c) phase structure grammar

Fig. 34.2: 3 simplified grammatical topologies

2002) spells out each subcategory *feature*, 'N(com, sing)'. ICE notation is more verbose, arguably more transparent, and features are more readily disassociated from each other than C5. As a result one can specify a query for a singular noun as 'N(sing)', rather than the wild card 'NN*1*'.

Table 34.1 lists a range of queries on an ICE POS-tagged corpus. Sentences are sequences of word/tag pairs, so queries might include simple strings; wild cards and logical expressions for words and tags; and sequences of these.

Tab. 34.1: A selection of typical POS-tag queries (ICE notation)

Query	applies to	description
fish fish* (expedition or trip) {expedition, trip, party} <N(sing)> <N or PRON> fish* + <N>	single lexical element	Simple string. Wild card. Logical expression of two simple strings. Set of alternative strings. Word-class tag. Logical expression of two POS tags. Wild card limited by POS tag.
fish* trip fish* (expedition or trip) <N> ? trip	strict lexical sequence	Wild card followed by simple string. Wild card followed by logical expression. POS tag, any lexical item, simple string.
fishing * trip	lexical sequence	Two simple strings separated by any number of intervening items.

2.2. A dependency grammar corpus

The topology of a typical dependency or constraint grammar is given in Figure 34.2(b). The string of tags in Figure 34.2(a) is extended by the introduction of a named link for each node, marked with an *f*, to another word/tag pair. (In some representations the final 'head word' may be linked to a further node.) These links form an acyclic tree where nodes cannot be linked in a loop.

The two visualisations in this figure are isomorphic. The *f* label, which we might call the *function* of the node or link, can be stored with the child node.

Dependency grammars have the following characteristics. A tree contains one node per word and no intermediate constituents. Some, but not all, dependency frameworks permit crossing links.

The most important point, of course, is that these analyses are not merely available to be admired or compared. Queries can exploit the tree.

Queries on a dependency grammar corpus must be able to express relations along the parent-child axis, marked 'grammar ordering' in Figure 34.2(b), as well as the sentence axis. In Table 34.1 the space between elements indicates that one word follows another, and '+' connects a word and tag. The presence of additional relational axes means we have to consider how these other relations are distinguished and expressed.

There are two broad traditions for expressing bundles, or *ensembles*, of relations in artificial intelligence. These are *logic-based* and *model-based* representations. In section 3 we discuss the relative benefits of each. But before we discuss how relations are put together in a query, let us first examine how the topology specifies a minimal set of individual relations.

We use the notation '**Relation**(*target*, *source*)' throughout this article, where '**Parent**(*c*, *a*)' means *c* is the parent of *a*. (Mathematical symbols are as follows: '$\wedge$' = and, '$\vee$' = or, '$\neg$' = not, '$\forall$' = for all, '$\exists$' = exists, '$\Rightarrow$' = entails, '$\Leftrightarrow$' = entail each other, '$\equiv$' = is defined as, '$\in$' = member of set, '{*a*, *b*}' = set containing *a* and *b*, '$\varnothing$' = unspecified.)

*a) **Parent***

In our example dependency grammar, the function *f* of a node is equivalent to a label associated with the link from the node to its parent. This has the implication that the function can be treated in a similar way to a feature of the node, and may be optionally included as a constraint in a query on that node.

If a grammatical vector is separable from its function, it is reasonable to elaborate an optimal set of general topological relations between two connected nodes in a query. The pair of diagrams in Figure 34.2(b) have the same topology, so the dependency vector between two word/node pairs (*a* and *c*) is equivalent to the relationship between a 'child' and 'parent' node. A query might express immediate or eventual connection along this axis, which we can denote as '**Parent**(*c*, *a*) ∈ {*Parent*, *Ancestor*}'.

In certain circumstances, parent-child directionality may be unimportant, so this set may include unordered relations, '*Parent/child*' and '*Same branch*'. Stating that '**Parent**(*c*, *a*). = *Parent/child*' would be equivalent to requiring that *a* and *c* were directly connected by a single arrow but that the direction was unspecified. Some small noun groups (or phrases) could be interpreted as having an ambiguous head, so within such a group it is conceivable that a tree may include a bi-directional arrow. Strict dependency is not obligatory in some constraint grammars. However, in the main, unordered **Parent** relations are usually *linguistically meaningless*. In a strict acyclic tree they will match two situations with radically different meanings: either *a* dominates *b* or *b* dominates *a*.

*b) **NextChild***

A second set of relations express the relationship between siblings, indicated by the dashed arrow in Figure 34.2(b). The relevance of this sibling axis depends on whether the order of children has meaning. Dependency grammars may not be restricted by word order. As we saw, some permit crossing links, in which case the actual order may be irrelevant.

If the grammar presumes that function labels specify an ordered set of 'slots' (e. g., subject, verb, direct object), but we do not know that elements *actually* appear in this order, the researcher may relax the ordering requirement. The issue for the linguist is whether an observed sequence is meaningful, i. e., it represents a phenomenon worthy of research.

An optimal set of relations are '**NextChild**(*b*, *a*) ∈ {*Next*, *After*, *Just before/after*, *Before/after*, *Branch after*, *Different branches*}'. The first four relate to sibling order and require that the two nodes share a common parent. '*Next*' and '*After*' state that the second node immediately, or eventually, follows the first in the sequence of children; the '*Before/after*' options are unordered. The last two values of **NextChild**, '*Branch after*' (ordered) and '*Different branches*' (unordered) refer to situations where two nodes might not share the same parent. This can occur if at least one child in the query has an '*Ancestor*' relation (Nelson/Wallis/Aarts 2002, 151−155).

If crossing links are not allowed in the grammar, '*Branch after*' is not required.

*c) **NextWord***

In a lexical or POS-tagged corpus, one may specify a query as a 'wild card' variation of a lexical stream, as in Table 34.1. We define the word-order relation set '**NextWord**(w_2, w_1) ∈ {*Next*, *After*, *Just before/after*, *Before/after*}' by analogy with **NextChild**.

d) HasNode

A final set of relations connects words and nodes. A POS-tagged corpus consists of a sequence of simple pairs of words and nodes, so a 'word plus tag' query (e. g. 'fish* + <N>') will suffice. In a parsed corpus, we may wish to express the concept that a node dominates a word, without requiring that they are directly connected. This distinction can be expressed as '**HasNode**(a, w) $\in$ {*Parent*, *Ancestor*}'.

In summary, to express queries on a corpus analysed with a dependency or constraint grammar, we need to be able to relate words and nodes together with four different types of relation, summarised by the sets **Parent**, **NextChild**, **NextWord** and **HasNode**.

2.3. A phrase structure grammar corpus

A phrase structure grammar adds nodes to those in dependency trees. These nodes represent phrases or clauses and bracket other sets of nodes (including single nodes) below them. A node can either tag a word or bracket a number of nodes (but not both). While a dependency tree has one node per word, a phrase structure tree contains up to twice as many (e. g., in ICE-GB the ratio is approximately 1.8). However, with this in mind, a phrase structure grammar and a dependency grammar have remarkably similar topologies. Compare Figures 34.2(b) and 34.2(c).

A phrase structure grammar is usually applied to a corpus in a descriptive rather than prescriptive manner. This has two possible implications. First, that a *strong ordering* restriction is applied, i. e. that the sentence sequence orders the tree, and prevents crossing links. Second, additional null (notional) words may not be inserted in the sentence, i. e. that the tree is *closed* by the sentence (see section 5.1.). If these topological restrictions are enforced, the query can be similarly constrained using axioms (see below).

For now, let us note that this similar topology means that essentially the same relation sets and axes − **Parent**, **NextChild**, **NextWord** and **HasNode** − are applicable to dependency and phrase structure grammars.

3. Representation, visualisation and matching

Earlier we made a distinction between matching and evaluation. At this point we discuss how relations are composed and visualised to form a query which can be matched against trees. We distinguish between formal *representation* − what a query may be composed of, and how constituents are integrated − and *visualisation*, which is concerned with how the query may be expressed and communicated. Naturally representation impacts on both visualisation and matching, and underpins the design of a search tool.

3.1. Criteria for evaluating query representations

How should we evaluate a query representation? On what basis should we prefer one representation over another? We propose the following set of criteria, in order of decreasing importance.

(i) **Linguistic adequacy.** It should be possible to express any query that has linguistic meaning. This is more important than absolute expressivity – just because one system is more formally expressive than another one does not mean that this expressivity necessarily has a *linguistic* benefit. For example, some query tools can state that two children have the same part of speech, without stating what it is. While expressive, this is of dubious linguistic value. Absolute expressivity may not be required for another reason. In a mature query platform one should be able to combine queries (typically in a logical expression). In summary, *the expressivity of a query system applied to a particular grammar circumscribes the set of linguistic concepts one can retrieve*.

(ii) **Transparency.** A more transparent representation is simply one easier to understand than another, given the same annotation scheme. The main problem that all users of parsed corpora face is sufficiently *learning the grammar* to achieve their goals. An important benefit of a transparent representation is that researchers can learn the grammar and *how it is applied to the corpus* by carrying out queries. Ideally, the user should be able to predict how a query matches examples in the corpus. The *totality* of the expression must be clear. We can see this in Table 34.1. Although lexical queries increase in complexity as one descends, there is a straightforward relationship between each expression and the cases it matches.

(iii) **Expressivity** is a formal property based on the expressivity of individual atoms and relations between them. Two representations are *representationally equivalent* if each can express everything the other can express. Sets ('{*a*, *b*}') and disjoint logical expressions ('$a \vee b$') are equally expressive. One representation is more expressive than another if a distinction can be made in the first that cannot be made in the second. Wild cards are more expressive than simple strings. The option to use unordered **NextChild/Word** relationships increases flexibility. Finally, the ability to relate nodes in a grammatical analysis represents further expressive possibilities.

(iv) **Efficiency** refers to the straightforward computational criterion of the implementation of a search. Although this is relatively unimportant compared to the ability to capture a linguistically meaningful expression, as corpora increase in size and complexity, retrieval efficiency is important in practice. Wallis/Nelson (2000) discuss this question in some detail.

The issues listed here are independent of a particular grammatical analysis. We have noted that different analysis schemes use different formalisms, encode different syntactic, morphological and semantic features, and are based upon different theoretical precepts regarding the meaning of terms. However, linguistic adequacy must ultimately be considered in relation to a particular analysis scheme. The corollary is that if one can identify the same linguistic phenomena in corpora annotated with different frameworks, the actual grammar deployed makes little difference to the results.

3.2. Logic-based queries

So far, we have elaborated four distinct classes of relation which queries must support. At this point we must make a basic decision as to how these relations are combined together. Traditionally there are two approaches: logic and models.

Advocates of a logic-based approach emphasise that logic is supremely expressive, yet despite this expressivity it retains a "clear formal semantics" (Hayes 1977). This means that a logical expression can be evaluated by a series of formal rules. Predicate logic is an extremely general formalism that may be used to express queries by the device of identifying elements and specifying relations between them.

To keep things simple, let us first review the use of logic in a POS-tagged corpus. The examples in Table 34.2 apply to two word sequences, w_1 and w_2, where w_1 immediately precedes w_2, represented by the **NextWord** predicate.

Tab. 34.2: Logical combinations of two-word queries

Query	description
1. $\exists w_1,w_2.(w_1 =$"fishing" $\wedge$ $w_2 =$ "trip" $\wedge$ **NextWord**(w_2, w_1))	Equivalent to "fishing trip".
2. $\exists w_1,w_2.((w_1 =$ "fishing" $\vee$ $w_2 =$ "trip") $\wedge$ **NextWord**(w_2, w_1))	Matches "fishing ?" or "? trip".
3. $\exists w_1,w_2.(w_1 =$ "fishing" $\vee$ $w_2 =$ "trip" $\vee$ **NextWord**(w_2, w_1))	Implausible. Matches "fishing", "trip" *or every word pair where* w_1 *precedes* w_2.
4. $\exists w_1,w_2.(w_1 =$ "fishing" $\wedge \neg (w_2 =$ "trip") $\wedge$ **NextWord**(w_2, w_1))	Matches "fishing ?" but not "fishing trip".
5. $\exists w_1,w_2.(w_1 =$ "fishing" $\wedge \neg (w_2 =$ "trip" $\wedge$ **NextWord**$(w_2, w_1)))$	Matches "fishing" where fishing is not followed by "trip".

Our first observation is that the expressivity of logic is at the expense of brevity. It is much simpler to write "fishing trip" than the equivalent logical expression. In Query 2, it is easier to list the two alternatives, *fishing* followed by a word ("fishing ?") or a word followed by *trip*, than work out the outcomes from the equivalent logical expression query. Query 3 is highly implausible and likely to be entered in error. It produces three different situations with limited linguistic connection. Permitting a relationship between elements to be optional is not very useful if the elements are left unrelated as a result.

The final pair of queries in Table 34.2 illustrates the importance of the scope of logical negation. The first matches *fishing* followed by any word, provided that this word is not *trip*. The second matches *fishing* except cases where it is followed by *trip*, in other words, it will also match cases where *fishing* is the last word in the sentence.

The problem with logic is twofold. Despite the phrase, 'a clear semantics' does not mean that an expression and its implications are readily understandable. It is easier to comprehend "[fishing ?] or [? trip]" than "$((w_1 =$ "fishing" $\vee$ $w_2 =$ "trip") $\wedge$ **NextWord**$(w_2, w_1))$". Moreover, as this example shows, provided that we allow queries to be *combined* with logic, many of the benefits disappear. We are left with queries like Query 3 in Table 34.2, where the expression specifies a set of alternative constraints on unconnected parts. Much of this expressivity does not seem to be linguistically very useful.

Most examples cited in favour of logic over models contain negation, e. g., 'find all clauses *without* a subject'. Such an expression is easy to understand but difficult to realise in a single conjoint model. However, one can achieve the same result by a process of subtracting the results of one query from those of another − remove the intersection of all clauses containing a subject from the set of all clauses. (In the introduction we distinguished between the evaluation and *organisation* of cases.)

What would a logical language sufficient for queries on a parsed corpus look like? *Tree Query Logic* (TQL, Wallis/Nelson 2000) and *Finite Structure Query* (*fsq*, Kepser 2003) implement queries in first order predicate logic. Below we use a TQL notation for consistency.

TQL employs a number of first order predicates that code for relations between two sorts of element in a tree. In section 2 we discussed a set of plausible topological relations between elements. We now consider how these might be translated into a logical formalism. Figure 34.3 illustrates a simple arrangement of three nodes, x, y and z, and two words, w_1 and w_2, which we will refer to in what follows. Table 34.3 lists a set of predicates sufficient to describe immediate and eventual relationships in a parsed corpus.

Tab. 34.3: Ordered TQL binary predicates

1-step predicate	multi-step predicate	axis	description
Parent(x, y)	**Ancestor**(x, y)	*parent:child*	x dominates y
NextChild(z, y)	**FollowingChild**(z, y)	*child:child*	z is after y in a sequence of children
HasNode(y, w_1)	**HasNodeAbove**(y, w_1)	*node:word*	node y annotates word w_1
NextWord(w_2, w_1)	**FollowingWord**(w_2, w_1)	*word:word*	w_2 is after w_1 in the sentence

Each predicate in the first column takes a single step in one direction along the equivalent axis. These are sufficient to construct a tree structure (or, conversely, a tree can be converted into a logical expression).

One can derive useful unary predicates. These *edges* of the query are indicated by grey 'T' marks in Figure 34.3.

$$\textbf{LastWord}(w_2) = \neg\exists w_3.\textbf{NextWord}(w_3, w_2). \quad \text{(there is no } w_3 \text{ following } w_2\text{)}$$

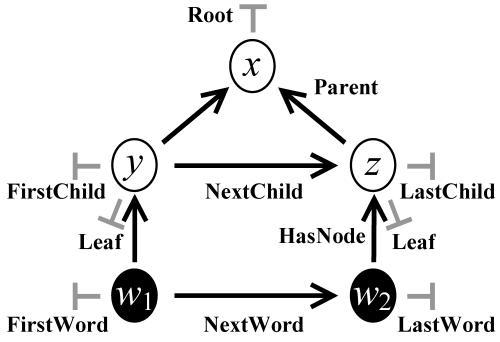


Fig. 34.3: Feasible predicates for a two-sort (word, node) TQL for a phrase structure grammar, after Wallis/Nelson (2000). In a dependency grammar x could also directly annotate a word

A similar process defines **FirstWord, Root, Leaf** etc.

Each single-step predicate can be complemented by another one (in the second column in Table 34.3) which covers an arbitrary number of steps. **FollowingWord** has the same effect as the spatial wild card '*'. If necessary one can derive unordered equivalents from ordered ones, but this is much more useful for model-based representations:

$$\textbf{AdjacentWord}(w_2, w_1) \equiv (\textbf{NextWord}(w_1, w_2) \vee \textbf{NextWord}(w_2, w_1)).$$

It is also necessary to introduce a number of topological axioms. These ensure that unfeasible trees cannot be constructed, including the following.

$$\textbf{Parent}(x, y) \Rightarrow \neg\, \textbf{Parent}(y, x) \quad \text{(circularity)}$$

$$\textbf{FollowingChild}(z, y) \Rightarrow \exists x.(\textbf{Parent}(x, y) \wedge \textbf{Parent}(x, z)). \quad \text{(children share a parent)}$$

Axioms may define the relationship between multiple and single-step predicates, thus:

$$\textbf{Parent}(x, y) \Rightarrow \text{Ancestor}(x, y).$$

Finally, topological properties of the grammar (see section 2) are expressed as axioms:

$$\textbf{FollowingChild}(z, y) \wedge \textbf{HasNodeAbove}(y, w_1) \wedge \textbf{HasNodeAbove}(z, w_2) \Leftrightarrow \textbf{FollowingWord}(w_2, w_1) \quad \text{(tree is ordered by sentence)}$$

$$\neg\exists x.\textbf{Parent}(y, x) \Leftrightarrow \exists w.\textbf{HasNode}(y, w). \quad \text{(tree is closed at words)}$$

We can complete the definition of relations outlined in section 2, and simplify some expressions as a result. However the fundamental problem is that in order to evaluate and comprehend a logical combination of relations one needs to draw a tree in the first place (if you needed to refer to Figure 34.3 to follow the argument above, you have proved my point). If this is the case, perhaps it is preferable to use query representations based on tree diagrams rather than logic.

3.3. Model-based queries

With the exception of *fsq* (Kepser 2003), model-based representations dominate this field. Although less theoretically expressive, models are simply easier to use. At the time of writing, for phrase structure corpora, they have been deployed by *tgrep* and *tgrep2* (Rohde 2001), *CorpusSearch* (Randall 2000), *LDB* (van Halteren/van den Heuvel 1990), *VIQTORYA* (Kallmeyer/Steiner 2003) and *ICECUP III* (Nelson/Wallis/Aarts 2002). Dependency grammar query tools include *TIGERsearch* (Lezius 2002) and *Netgraph* (Ondruška/Mírovský 2005). (See the papers for availability of software.)

The differences between the queries these tools can express are relatively minor, with much of the variation being in the grammar and how results are organised and evaluated.

To evaluate the benefits and costs of employing models to express queries, we will focus our discussion on one of these approaches, ICECUP's *Fuzzy Tree Fragments* (FTFs, Wallis/Nelson 2000) using examples based on the ICE phrase structure grammar, with trees drawn from left to right by default. However, the principles outlined here are common to all of these representations.

A model-based representation is one where elements are considered as part of a coherent whole. In logical terms this means that everything stated should be co-present, i. e., elements and relations are conjoined. Switching from logic to FTFs, therefore, loses negation and disjunction, except within prescribed limits. For example, ICECUP 3.1

(Nelson/Wallis/Aarts 2002) permits logical expressions within nodes and lexical items but only co-occurring combinations of relations. The result of this limitation is a much clearer representation. Let us consider an example.

Consider a query for a clause containing a noun phrase subject and direct object, sketched in Figure 34.4. In logic we would write something like the following:

$$\exists x,y,z.[\text{cat}(x)=\text{'CL'} \wedge (\text{cat}(y)=\text{'NP'} \wedge \text{func}(y)=\text{'SU'}) \wedge \text{func}(z)=\text{'OD'} \wedge \textbf{Parent}(x, y) \wedge \textbf{Parent}(x, z) \wedge (\textbf{FollowingChild}(y, z) \vee \textbf{FollowingChild}(z, y))]$$

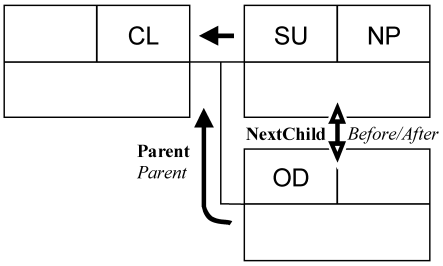


Fig. 34.4: A sketch of a query: a clause (CL) dominating a subject noun phrase (SU,NP) and a direct object (OD), in either order

The model-based representation guarantees a *coherent visualisation*. The total structure in Figure 34.4, representing a specific linguistic event, is immediately apparent and reflects tree structures in the corpus.

A Fuzzy Tree Fragment is a generalised grammatical subtree, containing notional node and word elements, with a series of named relational *links* and *edge* properties at each point. Edges are optionally specified as True or False. Links can take one of a number of values depending on the axis. **Parent** is mandatory and ordered: it must be either *Parent* or *Ancestor*. For the ICE grammar, **NextWord** and **NextChild** may be any of those listed in section 2 except *Branch after* (the grammar is strongly ordered by the sentence ordering).

Figure 34.3 shows how the applicability of particular links and edges depend on the configuration of nodes. Only the top-most node of a query, *x*, can be a **Root**. If two children are ordered by **NextChild**, the first, *y*, cannot be a **LastChild** and the second, z, cannot be a **FirstChild**. A similar principle applies to words.

ICE is a closed grammar and all leaf nodes have words attached. As a result the state of the **HasNode** relation may be deduced by the node's **Leaf** status.

$$\textbf{Leaf}(y) \Rightarrow \exists w_1.\textbf{HasNode}(y, w_1) \quad \text{(immediate)}$$

Conversely if it is possible that *y* is not a leaf, **HasNode** is eventual.

3.4. Visualising and matching queries

ICECUP (Nelson/Wallis/Aarts 2002) visualises FTFs by a system of colour-coded arrows, lines and edge markers. Against a grey background, black lines are used to depict the existence of an *immediate* connection **(Parent** in Figure 34.5) and white to indicate an *eventual* or *possible* connection. The absence of a line marks the impossibility of a

link. **NextChild** and NextWord relations are depicted by a system of arrows. In Figure 34.5, **NextChild** is *Before/After* (double headed white arrow) and NextWord is unspecified (no arrow). The various edges **(Root, Leaf** etc.) are drawn as white 'possible extensions' to the structure.

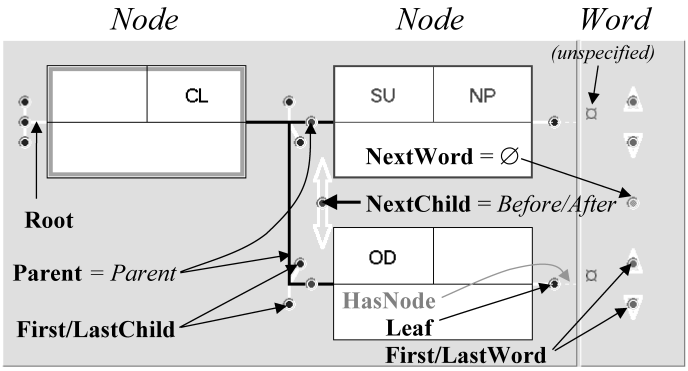


Fig. 34.5: An FTF visualised by ICECUP, (annotated), for the query in Figure 34.4

The benefits of a model-based approach should be immediately apparent. A tree-editing user interface, with extensions to set the status of links, can rapidly construct these fragments. FTFs are *visually coherent* − the total arrangement has meaning to the user − unlike a list of logical relations. A diagram is worth a thousand predicates. They are spatial structures rather than linear lists. This reflects the observation that tree diagrams are easier to follow than bracketed syntax (see, for instance, Quirk et al. 1985, 38). A diagrammatic approach is also readily extensible (see section 5) provided that complexity is carefully managed.

(As an aside, the value of graphical representations has been explored in a number of fields in science. Discussing the history of scientific creativity, for example, Cheng/Simon 1995 argue that selection of the 'right' diagrammatic representation has often proved to be the key to progress.)

The second advantage of employing tree models for queries on parsed corpora is that *it is much easier to visualise matching results*. Our query matches the ICE-GB treein Figure 34.6 twice.

1. [But [it]$_{SU,NP}$ needs [cooking]$_{OD}$ so we can see if it turns out all right]$_{CL}$
2. But it needs cooking [so [we]$_{SU,NP}$ can see [if it turns out all right]$_{OD}$]$_{CL}$ (ICE-GB S1A-012 #107)

We can illustrate these two matching patterns by colouring the nodes of a tree diagram. It is then easy to confirm how our query has matched the tree.

The FTF maps onto every matching tree in this way. Using a simple colour scheme, one can highlight each matching case in turn, and examine how different cases *interact* − for example, identifying that the second case above, *so we can see* ..., is subordinate to the first. This question of interaction is a general issue in corpus research (see below).

Examining matching patterns also reveals overlapping cases, as in the following tree (Figure 34.7).

3. The point is [[you]$_{SU,NP}$ can do [[what]$_{OD}$ [you]$_{SU,NP}$ like]$_{OD,CL}$]$_{CL}$ <laugh>

(ICE-GB S1B-007 #229)

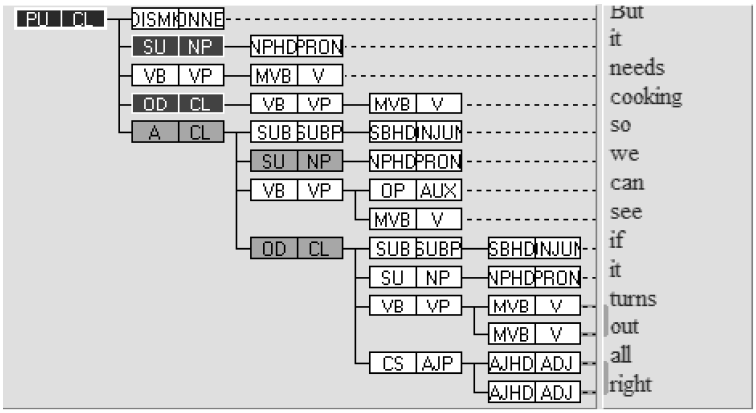


Fig. 34.6: Matching the same tree twice

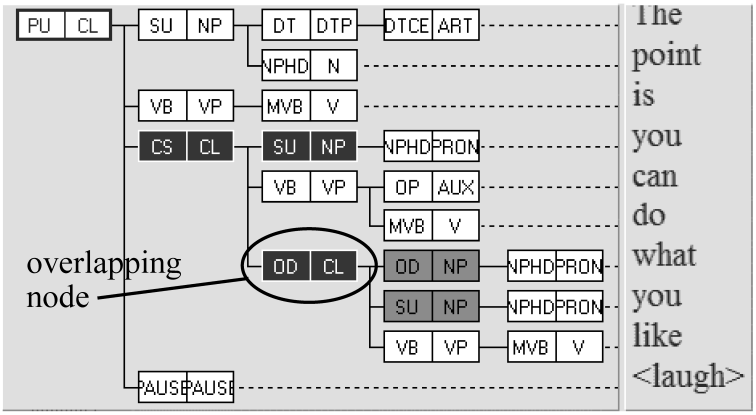


Fig. 34.7: Two cases sharing a node in an ICE-GB phrase structure tree

Here the direct object of the overarching clause *you can do what you like* is also a clause, *what you like*. One matching case is part of another. This example also illustrates the reversibility of constituents.

Although they are drawn as a tree, FTFs can be constructed to search for lexical sequences irrespective of the grammar (Nelson/Wallis/Aarts 2002, 147). The lexical sequence is linked by **NextWord** relations. The tree effectively disappears by attaching **Leaf** nodes, linked, via *Ancestor* relations, to a **Root** node. For every leaf, **NextChild** = ∅. Such an FTF will match once, unambiguously, to any tree annotating the lexical sequence.

ICECUP is an exploratory research platform that provides a 'forgiving' user interface that allows a researcher to construct a query and apply it to a corpus to obtain an exhaustive set of matching cases. The researcher can browse the set of matching cases and combine the results with those of other queries, e. g., by adding another, alternative FTF query, or by applying the results to a subcorpus.

Finally, the tree-model query representation permits one final trick. Tree model queries reflect trees in the corpus. They are *generalised tree structures*. One can *match* a query against a tree in the corpus. A user can also *abstract* a query from a tree or set of trees. Thus ICECUP's 'FTF Creation Wizard' tool constructs a query by taking a portion of a tree and converting it into a query.

4. Case interaction

Case overlap, as defined by a query, reflects a more general problem for researchers: *cases sampled from a corpus are not always independent*. Rather, with a few highly specialised exceptions aside, corpora consist of sequences of running text. Yet any process that *quantifies* results − from counting hits and calculating ratios to forming a dataset and testing hypotheses − effectively assumes that each case is independent from the next.

In fact, two cases in the same passage are less likely to be independent in origin than cases from different texts. Within a passage, two cases that are part of the same utterance are more likely to depend on each other than two more distant cases. Two cases in separate sentences are, on balance, less likely to interact than a pair within the same sentence. And finally cases within the same sentence interact to differing degrees − especially if their extent overlaps.

This problem has always existed in lexical corpora. Topic nouns and certain stylistic patterns obviously predominate in particular texts. In ICE-GB, modal *must* appears 4 times per 1000 words in eleven administrative/regulatory passages, but 0.65 times per 1000 across the corpus. Should each of these cases be treated as independent usages of equal worth, or are their uses due to a particular authorial style or subject matter?

In large lexical corpora such as the 100 million-word *British National Corpus* (Aston/Burnard 1998), a large number of sources and random subsampling can help to minimise the effect. In smaller parsed corpora, the grammatical evidence is both richer and rarer, and cases can appear in clusters for a variety of reasons.

We can consider case interaction as having two sources. The first is *conscious* repetition, including coordination or lexical choice, while the second is *grammatical*, where the use of one construction has consequences for subsequent constructions. Nelson/Wallis/Aarts (2002) offer the following range of possible sources of interaction in query results.

Full overlap. A case fully overlaps another. This is only possible if the query contains unordered relations (and one child can swap position with another).

Partial overlap. Part of one case coincides with part of another. There are two types:

(i) Two overlapping cases match *some of the same nodes* in the tree. This can arise if eventual relationships are employed in an FTF, such as '**NextChild** = *After*'.
(ii) Two cases overlap on different nodes, as in Figure 34.7, where the direct object of one match coincides with the head clause of a second. This is a type of embedding.

Embedding. One match can dominate or subsume another, e. g., a clause in a clause.

Coordination. Coordination normally comprises similar constructions, because one conjoin can usually replace the other or the entire coordinated structure.

Repetition. This occurs naturally for self-correction, reinforcement or stylistic reasons, within an utterance or in conversation.

Finally, it is well known that text genre and sociolinguistic context can lead to certain types of construction being preferred over others, e. g. interrogative clauses in interviews. A corpus sample should be *representative* of the population of utterances one is generalising about.

Strictly, any quantitative assessment of corpus data can only assume sampling independence if each case is sourced from a unique speaker and text. Since this requirement is often too restrictive in practice, an alternative is to try to quantify the *relative independence* of each case against the other cases in the same text. This may be defined as the probability that the case would arise *if the other cases in the text were absent*. To give this account some numbers: an independent case has a probability of 1, two explicitly repeated items within the same text, 0.5, and so on.

Grammatical independence can be estimated by a Bayesian method across the entire sample. If two cases a and b interact, the probability of b given a, $pr(b|a)$, is greater than the probability of b occurring independently, $pr(b)$. This increase in probability, $D(b, a) = pr(b|a) - pr(b)$, represents the dependence of b on a. This calculation must be generalised for all pairs of cases in the same text.

Case interaction is not critical in applications where quantified results are not required, such as obtaining examples for teaching or general exploration. However, this issue is increasingly important when carrying out linguistic research on a corpus. With simple experiments and relatively low-frequency close-interaction, one can 'downplay' the independence of cases, say, by underscoring χ^2 tests. More advanced solutions require the formal abstraction of an experimental model and the evaluation of an experimental sample, as discussed in Wallis/Nelson (2001).

5. Advanced topology

Up to this point we have only considered searching corpora with three types of grammatical topology. We have assumed that each word has a single node annotating it, and vice versa. We have assumed that coordination should be considered as part of the phrase structure. Finally, we have left aside how one might go about extending this basic topology to permit multiple tree analyses or to include other levels of analysis.

5.1. Compounds and missing words

Some parsing schemes do not assume that every word is given a node, but permit the use of compounds or 'ditto tags'. Some grammars, such as Treebank II (Marcus et al. 1994), permit the existence of nodes without words.

ICE treats both *to* and *in order to* as instances of 'particle *to*', adverbial *sort of* as a compound, and many titular proper nouns are analysed as compound nouns. This picture is further complicated by *discontinuous* compounds such as *be (just) going to*. Incidentally, Figure 34.6 shows two compounds, the phrasal verb *turns out* and adjective *all right*.

These situations represent a discontinuity between the grammar and the text. Grammatical relations treat compounds as single elements, while sentence relations assume every word within the compound to be distinct (Figure 34.8).

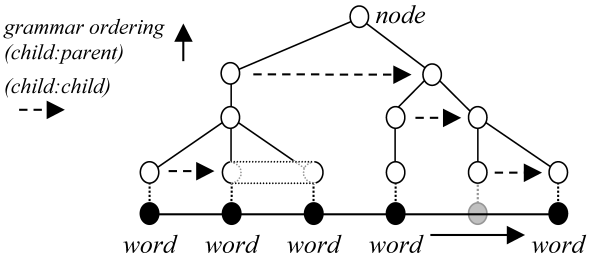


Fig. 34.8: Compounds and missing words

ICECUP solves this problem by employing a *late resolution strategy* (Wallis/Nelson 2000).

(i) Maximally **inflate the proof space** to place elements in every permissible position (including each position within a compound).
(ii) Apply the **proof** method to identify matching combinations of nodes and relations.
(iii) **Collapse the results**, carrying out any necessary simplification (e. g., two matching patterns which differ within the same compound may become one).

Null elements require a different solution. We explicitly increase the ambiguity of the query. Nulls can be accepted into the lexical sequence on the condition that the meaning of '**NextWord** = *Next*' is modified to match two words *separated by zero or more null words*. As a result the **NextWord** restriction may pair one word with several, and only by applying other restrictions is the ambiguity resolved.

ICECUP employs this method for skipping over elements such as pauses, punctuation and discourse markers that can appear at arbitrary locations in the lexical stream.

5.2. Coordination and self-correction

Coordination is not strictly part of phrase structure, but is better understood as being *tangential* to it. Coordinated elements, termed *conjoins*, can appear at many levels in a grammatical tree, from main clauses to verb phrases, e. g., *to swim and to fish*, adjective phrases, e. g., *high and mighty*, to adverbs as in *up and down*. An illustration is given in Figure 34.9.

Different analysis schemes can coordinate structures at slightly differing positions (ICE aims to coordinate phrases, where possible, while Treebank II will coordinate word classes). Coordination also entails a degree of explicit grammatical repetition. Moreover, one phrasal conjoin may contain an element, such as *d* in Figure 34.9, which is absent, but implicitly referred to, in another conjoin, as in the following from ICE-GB (S1A-002 #139).

4. I want to [[perform with a group]$_{CJ}$ and [do some choreography *for my final assessment*]$_{CJ}$]

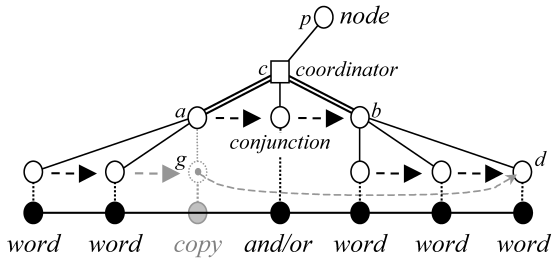


Fig. 34.9: Coordinating *a* and *b*, with gap $g = d$

In this *gapped conjunction* (Marcus et al. 1994), *d* is understood to be part of both phrases, *a* and *b*.

Each instance of coordination involves the insertion of a coordinating node into the tree that brackets conjoins and any intermediate conjunction (*and*, *or*, *but* etc.). The main problem for searching a corpus is that this arbitrary insertion of coordinating nodes into the structure (like the arbitrary insertion of discourse markers, see above) can be a confounding factor that prevents queries matching across the coordinator.

Suppose we have a query which looks for two nodes, *a*, *p*, where **Parent**(*p*, *a*). This will match cases without a coordinator but not those with the additional node. In order to permit the same query to match both coordinated and single phrases the 'immediate' **Parent** relation must be modified to match two nodes separated by zero or more coordinators. Any part of the query at *p* or above will match identical nodes and produce a partial overlap (see section 4 on case interaction).

Marcus et al. (1994) represent gapped conjunctions by introducing a new node (*g* in Figure 34.9) and creating a cross-referencing link $g = d$. Gaps can appear at any position in a conjoin, e. g., *Mary likes Bach and Susan, Beethoven*. Marcus et al. are attempting to recover predicate-argument semantics, '**like**(Mary, Bach) ∧ **like**(Susan, Beethoven)', whereas we are interested in (optionally) matching implied terms against full queries. As with null words and compounds (see 5.1.), the matching method first expands terms and then applies a late resolution strategy.

Every revision of the query system subtly modifies the meaning of the grammar. A query is an abstract tree. If coordination is treated differently from phrase structure in proof, then the meaning of coordinator nodes is distinct.

One of the biggest hurdles facing the parse analysis of spoken material is the identification and analysis of sections of text which are repeated and corrected dynamically by the speaker themselves. So-called *self-correction* is a well-known phenomenon in speech, and a research subject in its own right. However, during the parsing of corpora, self-correction is often only treated as a problem for the parser. ICE notionally 'removes' corrected material by setting an 'ignored' flag and adding surface annotation to show which areas of the text had been replaced by others. However, one interpretation of self-correction is two conjoins coordinated with 'or *rather*', as in:

5. ... the spectacle of seeing his older sister win [a prize]$_{CJ}$ − or, rather, [two prizes]$_{CJ}$

(ICE-GB W2B-006 #83)

The conclusion is that if coordinators can be made to 'disappear' in proof in the way suggested, self-correction could possibly be better represented as a form of *replacement coordination*, with a feature in the earlier material to indicate that it had been replaced.

5.3. Multiple analyses and levels

So far we have assumed that each sentence is given a single tree analysis. What if we permit multiple trees to be stored for the same sentence? We might do this for a number of reasons. Firstly, to represent *fundamental analytical ambiguity*, where additional analyses are (selectively) stored for ambiguous sentences only. Secondly, to represent *different analytical schemes* or *levels of analysis*. Thirdly, to represent *parallel translations*, where each translation is parsed separately. Below we discuss the implications of each of these, very briefly, in turn.

In the case of analytical ambiguity, one might represent an entire 'alternative' tree structure in the corpus, although the ambiguity might be confined to a small part of the tree. Consider a simple situation where a sentence has two interpretations. A query can match either tree independently. If it matches *both* trees, this counts as a single matching case. If it matches one tree, it should count as a single case *given the probability that this tree is the correct analysis*. It follows that each ambiguous tree should be given a prior probability of being correct (i. e. if equally plausible, 0.5 each), and the number of cases in each tree should be multiplied by this prior probability before being taken into account.

One benefit of corpora containing more than one parsing scheme is that they can be used for the contrastive evaluation of grammatical frameworks, *by comparing the effective retrievability of different representations*.

At the time of writing, the cost of multiple parsing has been too high for more than a microcorpus, the Amalgam MPC (see www.scs.leeds.ac.uk/amalgam/amalgam/multiparsed.html), to be constructed. Consider a corpus parsed with dependency and phrase structure grammars, as in Figure 34.10. We now have three types of element: words, plus two sorts of node (dependency and phrase structure), and two sets of structural relations that are applied to the two types of node.

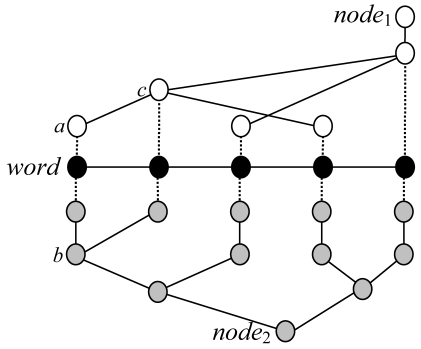


Fig. 34.10: A multi-parsed sentence

Queries on this type of corpus may now *span* the sentence and relate elements on either side together (such as *a* and *b*). In a parallel-parsed corpus, the sentence acts as the reference point for all grammars, allowing a single query to refer to any number of trees. The query representation for any tree necessarily employs the terms and axioms of the relevant grammar. In summary, the possibility of contrasting grammars in this way has two requirements: (a) the definition of a particular evaluative measure (retrievability,

i. e., what can be identified or derived from the structure), and (b) the fact that all trees are grounded in the same set of sentences.

A similar principle applies to the integration of a supplementary *morphological* layer of analysis into a parsed corpus. The most integrated representation is where the same topology is carried through below the level of the tree, so a constituent analysis is preferable with a phrase structure grammar, or a dependency analysis with a constraint grammar.

Adding a morphological analysis of each word in a parsed corpus, and extending grammatical queries accordingly, permits new avenues of research. One could investigate the interaction of morphology and syntax, and evaluate morphological dependencies from first principles within grammatical constraints (e. g. within the same phrase or clause.)

Provided that morphological terms can be represented within the same framework as the grammar, we could expect to see networks like Figure 34.11, consisting of an acyclic tree where words are no longer terminals and where morpheme sequences subdivide words. In practice it may be desirable to further extend the annotation to represent morphological compounds, relate morphemes to the lexical string (e. g. indexing a character range in the word) and directly relate morphemes to words.

Corpus queries would then include an explicit **HasNode**(*t*, *w*) relation for words and morphemes and relations along the morpheme axis. The interpretation of **NextMorpheme** would be limited, unless otherwise stated, to a sequence of morphemes *within* a word. Thus, to follow the marked arrows in Figure 34.11, one would have to consider two *Ancestors* with attached words, on either side of a **NextWord** relation. (Naturally, it is easier to construct this model than to summarise it.)

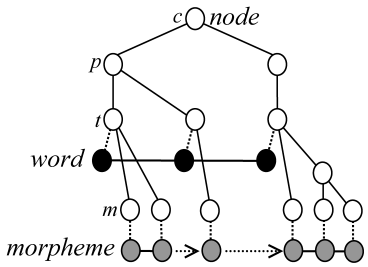


Fig. 34.11: From morpheme to clause: a morpho-syntatic tree

The final corpus typology we will consider is that of parsed translation corpora, where an L1 (first language) sentence is translated into an L2 (second language) sentence and both sentences are parsed. In a phrase structure grammar, one can then map L1 nodes, including phrases and clauses, to L2 nodes (Figure 34.12).

Word-for-word machine translations have well known problems (cf article 32). A solution, in theory at least, is to parse sentences and carry out translation at a phrasal level. An important motivation for carrying out research on this kind of corpus is to identify regular translation patterns and rules.

Although the same grammar may be used for each language, queries are either applied to one or the other language, or form a composite pair linked by the single-place translation arc.

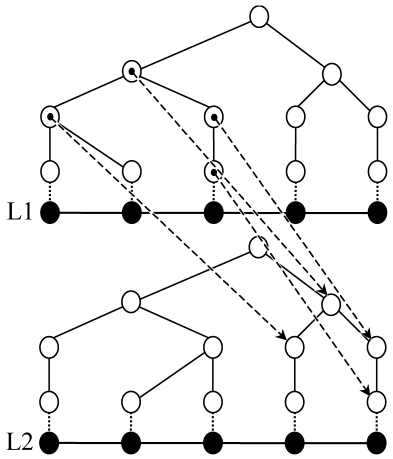


Fig. 34.12: Mapping parsed translations

6. Conclusions

The degree to which a parsed corpus may be exploited depends on the effective use of queries. A clear visual query representation is essential for linguistically educated users and researchers. Model-based queries convey a diagram of the desired total structure, but sacrifice the absolute expressivity of logic. The formal expressivity of logic is often of limited value, however. It may be misleading, and the result is a query platform that is difficult to use by non-specialists. Moreover, a focus on formal expressivity ignores the fact that users must learn the application and meaning of the grammatical annotation in order to carry out research on a parsed corpus.

By contrast, tree models like *Fuzzy Tree Fragments* are cohesive and have an intuitive appeal, the manner in which they have matched against trees in the corpus can be understood, and they may be abstracted from corpus trees. Tools for parsed corpora employ a number of different query systems, with an emerging consensus around the use of treelike models. ICECUP and its brethren have been used for a variety of linguistic tasks, from teaching and research to intensive parse correction (Wallis 2003).

Some of these tools simply concern themselves with retrieving matching cases, although the more mature support an entire exercise of exploring some aspect of the corpus and carrying out simple experiments.

At this point we note that effective experimental research with parsed corpora should take into account the fact that one linguistic event may interact with another. An emerging area of research is in developing experimental methodologies that both incorporate the entire experimental process and automate it as much as possible. An *experimental environment* (Wallis/Nelson 2001) transforms a corpus from being a source of inspiration and statistical distributions to a *locus of theoretical discussion*, where general concepts in linguistics may be evaluated against corpora shared by the entire community, and where interpretations of the results of studies may be widely discussed.

Finally, we discussed some of the implications of more complex corpus topologies, from specific variations on a parsed corpus to the representation of multiple analyses. Notwithstanding our inability to anticipate future developments in corpus linguistics, it seems that model-based 'tree fragment' queries are here to stay.

7. Acknowledgements

Thanks to Dick Hudson, Evelien Keizer, Rolf Kreyer, Anke Lüdeling, Danny Mukherjee, Gerry Nelson, Joakim Nivre, Gabriel Ozon and an anonymous reviewer for their helpful comments on this paper. Initial research was carried out under ESRC grant R000222598. ICECUP III is available from www.ucl.ac.uk/english-usage. Over the years I have been privileged to gain insights from numerous users of ICECUP and to participate in fraternal discussions with authors of other query systems. Every query system is dependent on the quality and reliability of parse analyses, so it is to the unsung galley slaves of corpus annotation that ultimate thanks are due.

8. Literature

Abeillé, A. (ed.) (2003), *Treebanks: Building and Using Parsed Corpora*. Dordrecht: Kluwer.

Aston, G./Burnard, L. (1998), *The BNC Handbook: Exploring the British National Corpus with SARA*. Edinburgh: Edinburgh University Press.

Briscoe, T. (1996), Robust Parsing. In: Cole, R. A./Mariani, J./Uszkoreit, H./Zaenen, A./Zoe, V. (eds.), *Survey of the State of the Art in Human Language Technology*. <http://cslu.cse.ogi.edu/H LTsurvey/ch3n ode9.html>.

Cheng, P. C.-H./Simon, H. A. (1995), Scientific Discovery and Creative Reasoning with Diagrams. In: Smith S./Ward, T./Finke, R. (eds.), *The Creative Cognition Approach*. Cambridge, MA: MIT Press, 205−228.

van Halteren, H./van den Heuvel, T. (1990), *Linguistic Exploitation of Syntactic Databases: The Use of the Nijmegen Linguistic DataBase Program*. Amsterdam: Rodopi.

Hayes, P. J. (1977), In Defence of Logic. In: *Proceedings of the 5th International Joint Conference on AI (IJCAI-5)*, Cambridge, MA, 559−565.

Järvinen, T. (2003), Bank of English and beyond. In: Abeillé 2003, 43−59.

Kallmeyer, L./Steiner, I. (2003), Querying Treebanks of Spontaneous Speech with VIQTORYA. In: *Traitement Automatique des Langues* 43(2), 155−179.

Karlsson, F./Voutilainen, A./Heikkilä, J./Anttila, A. (eds.) (1995), *Constraint Grammar: A Language-independent System for Parsing Unrestricted Text*. (Natural Language Processing 4.) Berlin/New York: Mouton de Gruyter.

Kepser, S. (2003), Finite Structure Query: A Tool for Querying Syntactically Annotated Corpora. In: Copestake, C./Hajič, J. (eds.), *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2003*. Budapest, Hungary, 179−186.

Lezius, W. (2002), TIGERSearch − Ein Suchwerkzeug für Baumbanken. In: Busemann, S. (ed.), *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002)*. Saarbrücken, Germany, 107−114. See also <http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch>

Marcus, M./Kim, G./Marcinkiewicz, M. A./MacIntyre, R./Bies, M./Ferguson, M./Katz, K./Schasberger, B. (1994), The Penn Treebank: Annotating Predicate Argument Structure. In: *Proceed-*

ings of the Human Language Technology Workshop. San Francisco: Morgan Kaufmann, 114–119.

Marcus, M./Santorini, B./Marcinkiewicz, M. A. (1993), Building a Large Annotated Corpus of English: The Penn Treebank. In: *Computational Linguistics* 19(2), 313–330.

McEnery, T./Wilson, A. (2001), *Corpus Linguistics*. 2nd ed. Edinburgh: Edinburgh University Press.

Nelson, G./Wallis, S. A./Aarts, B. (2002), *Exploring Natural Language: Working with the British Component of the International Corpus of English*. (Varieties of English around the World.) Amsterdam: John Benjamins.

Ondruška, R./Mírovský, J. (2005), *Netgraph Client Manual*. Institute of Formal and Applied Linguistics, Prague. <http://quest.ms.mff.cuni.cz/netgraph/doc/netgraph_manual.html>.

Quirk, R./Greenbaum, S./Leech, G./Svartvik, J. (1985), *A Comprehensive Grammar of the English Language*. London: Longman.

Randall, B. (2000), *CorpusSearch User's Manual*. Technical report, University of Pennsylvania. <http://corpussearch.sourceforge.net>.

Rohde, D. (2001), *Tgrep2*. Technical report, Carnegie Mellon University. <http://tedlab.mit.edu/dr/Tgrep2>.

Wallis, S. A. (2003), Completing Parsed Corpora: From Correction to Evolution. In: Abeillé 2003, 61–71.

Wallis, S. A./Nelson, G. (2000), Exploiting Fuzzy Tree Fragments in the Investigation of Parsed Corpora. In: *Literary and Linguistic Computing* 15(3), 339–361.

Wallis, S. A./Nelson, G. (2001), Knowledge Discovery in Grammatically Analysed Corpora. In: *Data Mining and Knowledge Discovery* 5(4), 305–336.

Sean Wallis, London (UK)