Data Mining and Knowledge Discovery, 5, 305–335, 2001

Knowledge Discovery in Grammatically Analysed Corpora

SEAN WALLIS* s.wallis@ucl.ac.uk
University of HongKong, Department of English, HongKong

GERALD NELSON ganelson@hkucc.hk
Survey of English Usage, University College, London, UK

Editors: Fayyad, Mannila, RamaKrishnan

Received September 14, 1999; Revised June 9, 2000

Abstract. Collections of grammatically annotated texts (corpora), and in particular, *parsed* corpora, present a challenge to current methods of analysis. Such corpora are large and highly structured heterogeneous data sources. In this paper we briefly describe the parsed one-million word ICE-GB corpus, and the ICECUP query system. We then consider the application of *knowledge discovery* in databases (KDD) to text corpora. Following Cupit and Shadbolt (Proceedings 9th European Knowledge Acquisition Workshop, EKAW '96; Berlin: Springer Verlag, pp. 245–261, 1996), we argue that effective linguistic knowledge discovery must be based on a process of *redescription* or, more precisely, *abstraction*, based on the research question to be investigated. Abstraction maps relevant elements from the corpus to an abstract model of the research topic. This mapping may be implemented using a grammatical query representation such as ICECUP's *Fuzzy Tree Fragments* (FTFs). Since this abstractive process must be both experimental and expert-guided, ultimately a workbench is necessary to maintain, evaluate and refine the abstract model. We conclude with a pilot study, employing our approach, into aspects of noun phrase postmodifying clause structure. The data is analysed using the UNIT machine learning algorithm to search for significant interactions between domain variables. We show that our results are commensurable with those published in the linguistics literature, and discuss how the methodology may be improved.

Keywords: linguistics, grammar, structured datasets, Text Corpora, redescription, cyclic knowledge discovery

1. Introduction

Corpus linguistics attempts to gain linguistic knowledge through the analysis of collections of samples of naturally-occurring texts and transcribed recordings. Corpora are composed of selections of material, usually of a normalised extent, taken from a variety of written and spoken genres. Material may be sampled over time, geography or language. Corpus texts (henceforth we take 'texts' to include transcriptions) are usually *annotated* by augmenting

*The UNIT machine learning tool was developed by the first author at the AI Group, University of Nottingham, where it was supported by the REPAY project (BRITE/EURAM 4139). The first author was supported by UK ESRC grant R000222598 to develop the FTF query system. The construction and annotation of the ICE-GB corpus was supported by ESRC grant R000232077.

further levels of description that illustrate specific aspects of language production. In particular, it is common to provide some kind of *grammatical* annotation.

At a minimum, this consists of an individual grammatical classification for every lexical item in the corpus, by labelling words with wordclass tags (*noun, verb*, etc.), possibly with additional features (*proper, singular*, etc.). This process is relatively simple and can be performed automatically by stochastic methods to around 95% accuracy (Brill, 1992). As a result very large tagged corpora have been produced. For example, the *British National Corpus* (BNC; Burnage and Dunlop, 1992) is a 100 million word tagged corpus of (largely contemporary) British English. (See http://info.ox.ac.uk/bnc) By contrast, automatically *parsing* natural unrestricted text will only have a partial success (Briscoe, 1996). In practice, therefore, results must be checked, corrected and cross-checked by trained linguists.

Texts might also include prosodic, phonetic, dialogue or semantic annotation. Multilingual *translational* corpora have been produced that include text translations. Depending on the translation, an attempt at mapping between the two versions of the text may be made, forming a further, intermediate layer of annotation. However, like parsing, adding each class of annotation requires human expertise.

Annotated corpora are expensive to construct and relatively rare. Effective exploitation is therefore a priority. The selection of material, corpus size, and choice of annotation levels are determined by linguistic goals and perspectives. This implies that, for general application, a broad selection of material and a detailed level of annotation is required. Recently constructed corpora are beginning to address this requirement. *The University of Pennsylvania Treebank* (UPenn Treebank, Marcus et al., 1993) is a grammatically parsed corpus of around 2.5 million words of (mainly written) US English, while the *British Component of the International Corpus of English* (ICE-GB, see Section 2 below) contains one million words of spoken and written British English. Moreover, attention is now being directed towards languages other than English. A workshop in 1999 (Abeille, 1999) discussed work in progress in producing parsed corpora of comparable size in seven further languages.

Tagged corpora have been used to construct word lists for dictionaries and thesauri. Techniques for studying these corpora have focused on counting frequencies and exemplification (McEnery and Wilson, 1996). Parsed corpora, on the other hand, contain an analysis of sentence structure in the form of a single unambiguous grammatical *tree analysis* for every sentence in the corpus (see figure 1).

This paper addresses a new prospect that arises from the development of these detailed parsed corpora (and the revolution of mass cheap computing power). How can linguists exploit parsed corpora in order to carry out research into the way that language behaves in 'real' use? Moreover, what tools and techniques are appropriate? Can linguistic knowledge be encoded in such a way as to effectively direct computational power? Can new linguistic insights be gained through the application of search? Can this knowledge be shared, transferred and reproduced among a community of linguists in such a way as to promote debate about theoretical principles underlying differing grammatical perspectives?

Linguistics has long been divided between corpus based and theory driven approaches. However, in this paper we will argue that knowledge-guided analysis of parsed corpora can act as a bridge between linguistic theory and corpus data. The challenge is to use the corpus

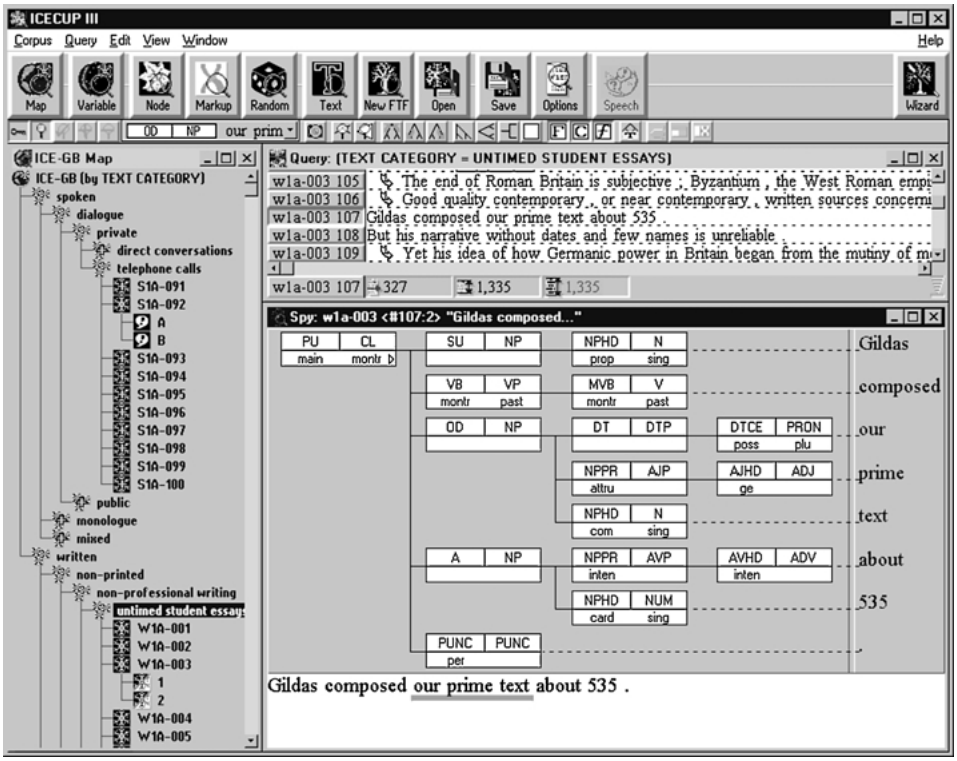


Figure 1. Three levels of corpus description: *corpus* (left), *text sequence* (top right), and *single unit* (with grammatical analysis, bottom right).

to evaluate, test, and generate hypotheses that are commensurable with those of traditional structural linguistics.

Much of the impetus for constructing parsed corpora, particularly in the US, has been to provide a realistic training set for automatically generalising (learning) parsing systems. Corpora such as the UPenn Treebank are primarily used as training and benchmark resources for parsing algorithms.

However, the construction of performance systems, such as learning parsers, does not promise a shortcut to meaningful linguistic knowledge. In practice, such systems are evaluated by performance criteria rather than by the much more nebulous notion of 'meaning'. A 'better' parser is one that is judged to produce a more accurate parsed tree for each sentence in a text. There is a trade-off here, due to the complexity of natural language and the law of diminishing returns.[1] Given the same background knowledge, the more accurate an inductive parser, the less comprehensible its internal representation will tend to be. Conversely, obtaining meaning requires first *introducing background knowledge* into the generalisation process. In fact, the most successful parsers have been hybrid systems that use a combination of expert-supplied rules and corpus-derived statistics.

In contrast with this 'hard', automated, approach to corpus linguistics, many linguists have maintained an exploratory, investigative, 'soft' approach. Soft corpus linguistics requires computation to select material from a corpus, but does not aim at autonomous discovery. In this paper we describe the initiation of a new research programme: to apply *knowledge discovery in databases* (KDD) to text corpora. Our approach, although involving significant computation, does not remove the linguist from the discovery process. Rather, it aims to *empower* linguists with the necessary computational support to investigate the strength of theoretical claims and—using *machine learning* (ML) techinques—to permit the search for new, linguistically meaningful results.

2. ICE-GB and ICECUP

The Survey of English Usage recently published a one-million word corpus of contemporary British English, the *British Component of the International Corpus of English* (ICE-GB; Greenbaum, 1996a, b). ICE-GB is part of the worldwide International Corpus of English (ICE) project, which aims to collect and annotate corpora on identical lines for some 20 native varieties of English. Supplied to linguists with the corpus is *ICECUP III* (Wallis, and Nelson, 2000; Nelson, Wallis, and Aarts, in press), a program that provides an integrated platform for the exploration of parsed corpora.[2] This provides a variety of *query systems*, both sociolinguistic and grammatical, on a specialised *corpus management system* (CMS) platform. Grammatical queries are expressed as *Fuzzy Tree Fragments* (FTFs), which are described in detail in (Wallis and Nelson, 2000) and summarised in Section 3 below.

ICE-GB is unusual among parsed corpora in that it contains a majority (60%) of orthographically transcribed spoken English. The fundamental division in the principal sampling variable, called 'text category', is spoken versus written, which is further subdivided (figure 1). This classification hierarchy is augmented by a variety of sociolinguistic and biographical data organised by text, subtext, and each speaker or author.

Sociolinguistic variables are secondary, however, to the sheer detail of the textual analysis. ICE-GB contains 500 2,000-word texts and over 83,000 grammatically analysed text units. The level of grammatical detail in ICE-GB is far in excess of the UPenn Treebank. In grammatical terms, we obtain a *complete* parse for all sentences in the corpus, whilst Marcus' group use a skeleton parsing scheme (see Section 7.1). Analyses in ICE-GB have been checked and cross-checked by linguists using ICECUP. We estimate that the entire effort was of the order of 25 person years.[3]

Aside from the grammatical analysis, the corpus is annotated in other ways. In order to consider them grammatically, transcriptions are first edited into approximate 'sentences', or *text units*, marked by their speaker. Annotation indicates where speakers overlap one another, correct themselves, or break off in mid-utterance. Written texts include mark-up for typographic details and editorial (spelling) normalisations. Figure 1 illustrates these three principal levels of description in ICE-GB and their realisation.

The corpus is fully indexed by atomic elements and sociolinguistic variables. In many circumstances, the software can perform rapid queries by simply retrieving an index of all matching cases. For complex structured queries, however, it is necessary to determine the

correct relationship between components. Candidate text units must be *identified, retrieved*, and *matched* by a proof method (Wallis and Nelson, 2000).

3. Fuzzy tree fragments

Fuzzy Tree Fragments (FTFs: Aarts, Nelson, and Wallis, 1998; Wallis and Nelson, 2000) are a representation for structured grammatical queries.[4] FTFs are *generalised grammatical subtrees* representing a model of the grammatical structure sought, with only the essential elements retained—a 'wild-card' model for grammar. The idea is fairly intuitive to linguists while retaining a high degree of flexibility. Nodes and text unit elements may be approximately specified, as may relational links between elements, and unary structural properties termed 'edges' (e.g., 'x is the first child in a sequence').

FTFs are diagrammatic representations: they make sense as drawings of partial trees rather than as a set of logical predicates. Such diagrams have *structural coherence*, that is, it is immediately apparent if an FTF is feasible and sufficient (grammatically and structurally).

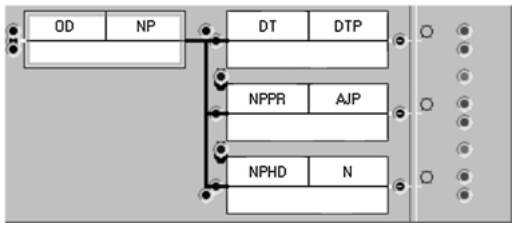
Links are coded for *adjacency, order* and *connectedness*, and depicted so as to exploit structural coherence. Thus, the *parent*:*child* relation in an FTF can be either immediately or eventually adjacent, called 'parent' and 'ancestor' respectively, and coloured black or white. On the other hand, the sibling *child*:*child* relation may be set to one of a number of options, from 'immediately following' (depicted by a black directional arrow), through 'before or after' (a white bi-directional arrow), to 'unknown' (no arrow). The latter is necessary if two 'child' nodes in an FTF are not necessarily siblings in the corpus tree. A full list of all links and edges is provided in (Nelson, Wallis, and Aarts, in press).

The final benefit of the graphical approach is that, as we shall see (figure 3), it is relatively easy to see the relationship between an FTF and trees in the corpus. This applies both to *matching* and to *abstraction* (creating an FTF from an example in the corpus).

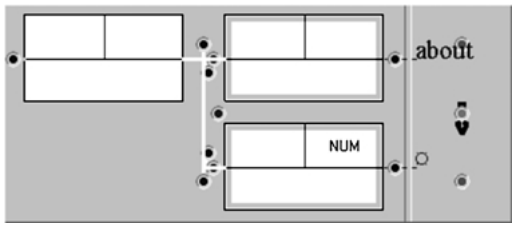
Parsed text units in a corpus may be considered as either (a) contiguous *text* with attached grammatical annotation, or (b) banks of *trees* containing embedded text. FTFs can be used to search for similar text sequences regardless of grammar, similar grammatical structure regardless of text, or combinations of grammar and text.

The examples in figure 2 illustrate this point. In (2a), the nodes are immediately adjacent and must appear in precisely this arrangement in a tree. No other information is specified. Example (2b) expresses a "word, tag" sequence as an FTF. The two nodes on the right are bound to the leaf position, so they directly annotate the text unit elements to their right. The third node matches the root of the tree and is connected to the leaf nodes by eventual 'ancestor' links. There is no sibling relation between the leaves, but the text unit elements are ordered by the black arrow on the right, which specifies that the second, unspecified, word immediately follows "about" in the sentence.

All FTFs contain a *focal point*, which we currently define as a single node or set of sibling nodes in the tree. The focal point allows us to determine independent cases. If we wish to study *direct object noun phrases* (written '[OD, NP]'), for example, we can locate the focus in the FTF on this node (indicated by the border on the leftmost node in figure 2a). ICECUP employs the focal point to indicate the portion of text 'covered by' the FTF, and to organise *concordancing* displays (Nelson, Wallis, and Aarts, in press).



(a)



(b)

Figure 2. FTFs for grammatical and textual queries. (a) Grammatical query for a type of direct object (OD) noun phrase (NP).[5] (b) Textual query for a word + tag sequence: "about" and a numeral.

Figure 3 indicates how the two FTFs match nodes in the text unit in figure 1. The text-oriented FTF, "about <NUM>" (2b) matches the leaf nodes '[AVHD, ADV]' and '[NPHD, NUM]' in the centre of the figure. The tree-oriented FTF (figure 2a) matches the four nodes in the middle. The focal point of the former encloses the two leaf nodes and thus the lexical items "about" and "535". The focal point of the latter encloses the entire *noun phrase* (henceforth: NP), realised by "our prime text". These foci are illustrated in the lower portion of the window.

Note that an FTF may multiply-match the same tree (text unit) in a number of ways. Firstly, an FTF may match *distinct parts of the tree*. For example, figure 3 contains three noun phrases, so a search for all NPs would find three separate matches in this case. Secondly, FTFs may match overlapping portions of the same tree. This is exacerbated by the ability to specify inexact relationships between elements in an FTF. Finally, an FTF can include unordered sibling and word relationships, so it may match the same set of nodes more than once.

The topological issue of distinguishing between matches leads to a more general ontological question: *what is an independent case in the corpus?* A 'case' in a study of NPs, for example, will necessarily be a fraction of a tree. Cases can overlap or subsume one another (e.g., NPs within NPs). This presents a problem in statistical analysis, as strict sampling assumptions of independence are weakened. We return to this problem in Section 4.3 below.

ICECUP permits a linguist to explore the parsed corpus quickly and—given the annotation complexity—relatively intuitively. A 'Wizard' tool supports the abstraction of an FTF from a tree in the corpus, a 'find me something like *that*' facility. The software employs an *exploratory methodology* (Wallis, Nelson, and Aarts, 1999) based on the assumption

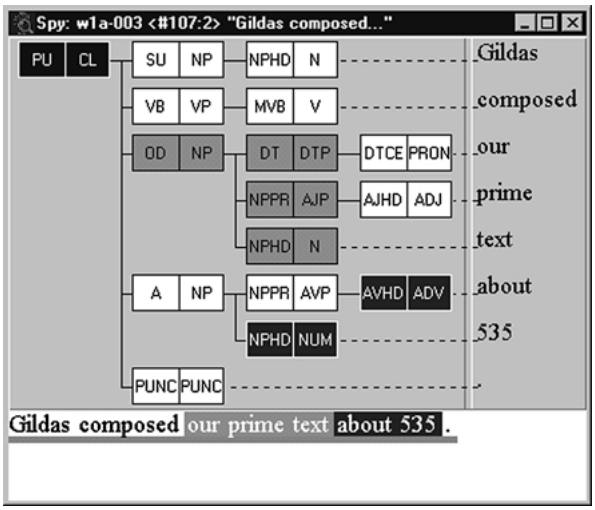


Figure 3. Matching FTFs in figure 2 against a single tree.[6]

of incomplete prior user knowledge. Even experienced linguists cannot know, in advance, every detail of the grammatical representation employed in a corpus, or how it is realised.

However, whilst investigations carried out by performing queries and examining results may uncover results, the exploration process is heavily reliant on the researcher, and consequently, is time consuming and limited. The mode of research is constrained by (a) the necessity to specify each individual query, and (b) weak control over the process. These exacerbate the effect of *a priori* biases. We argue that to fully linguistically exploit a corpus one must provide a different kind of exploratory system, one that encompasses a mini 'research programme' (Lakatos, 1970) rather than a single research question.

4. Knowledge discovery in corpora

4.1. The 3A perspective on corpus linguistics

'Soft' linguistic research on text corpora (see Section 1) may be characterised as three distinct stages, what we call the '3A perspective' (*Annotation—Abstraction—Analysis*, figure 4). Each of these stages represent a potential source of error between the original text and the evaluation of hypotheses. Each is knowledge intensive and approximate, and therefore cyclic.

Note that the statement that each process requires the introduction of further knowledge entails that it is not possible to generate equivalent results through autonomous processing. 'Hard' corpus linguistics is limited *in principle* because it eschews the acquisition of expertise for algorithmic refinement.

Our current ICECUP software supports the first two stages. As we mentioned, it was used to cross-check and correct the grammatical annotation in ICE-GB. Secondly, ICECUP

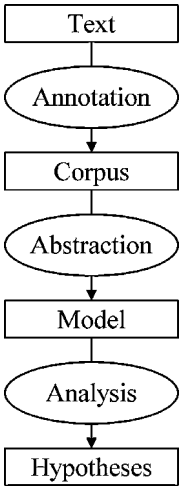


Figure 4. The 3A perspective: from text to hypothesis.

permits exploratory access to the annotated corpus, and offers a variety of query systems to support abstraction. It also employs a propositional calculus to combine queries (Nelson, Wallis, and Aarts, in press), so that a linguist can compare how a grammatical structure is used in formal and informal writing, for example. However, comparison is primarily visual rather than analytic.

Statistics, such as χ^2 tests, may be applied to the results,[7] but the principal analytic limitation derives from the fact that ICECUP does not currently support the construction and manipulation of an explicit *domain model*.

Section 5 describes a pilot experiment applying redescriptive knowledge discovery to a corpus, exploiting the existing FTF query system. We employ a machine learning algorithm, UNIT (Wallis, 1993; 1999b; see Section 5.4), which obtains hypotheses in the form of sound *independent rules* and exploits *hierarchical domain variable* structure.[8]

4.2. A perspective for discovery

Knowledge Discovery in Databases (KDD) has emerged out of ML and other statistical techniques to become an established discipline in its own right. 'Machine learning' itself comprises a diverse set of computational techniques whose shared characteristic is that they attempt to inductively establish trends and interactions in data. In a KDD context, the idea is to employ learning systems to detect significant interactions that may require further explanation and exploration. ML is not a substitute for a deep analysis, but a tool to identify significant interactions. The main advantages of a statistically sound *learning* approach over more traditional statistical methods are search depth and the consequent tendency to reduce *a priori* researcher bias.

However, often a regular database is not supplied in the appropriate form to permit theoretically interesting results to be obtained with an algorithm. Simply applying an ML

algorithm to an existing database, pejoratively termed 'data mining', has had limited success (Fayyad, 1997). Cupit and Shadbolt (1996) argue that this is because fields are characterised by the requirements of *data collection*, whereas KDD is employed *post-hoc*, often for quite distinct purposes. Related to this is the question of domain structure. Databases often include fields that are not strictly independent from one another. For example, in a medical survey, if a patient is male, the question, *have you ever been pregnant?* is superfluous. Blind application of a statistical method is then liable to detect these strong but trivial interactions.

Effective learning, therefore, requires a database that is, at minimum, *normalised* by the elimination of duplicate, dependent and irrelevant fields (Wallis, 1993). Cupit and Shadbolt argue further, recommending that the data should be *redescribed in terms of the research question to be solved*. Their CASTLE workbench permits records in a source database to be mapped to cases in an abstract dataset, or *domain model*, defined by the requirements of the research. This translation is defined in two parts: a set of *domain variables*, and, for each variable and value, a *mapping rule* that specifies the conditions when this particular value is defined. A deterministic translation process is then applied to obtain the domain model, which may then be explored using ML and statistical methods. Analytic results may be used to further refine the mapping, in a cyclic process.

Naturally, this approach is *knowledge dependent*. The results of analysis are dependent on this domain characterisation. A principled *knowledge acquisition* (KA) process, therefore, must be employed to guide the specification of variables and mapping rules.

4.3. The necessity of redescription in corpus-based analysis

Cupit and Shadbolt's characterisation of KDD as being centred around research-oriented redescription is central to 'knowledge discovery in corpora' (KDC). While in some domains, simple 'data mining' is conceivable, in the case of text corpora (and, possibly, for other 'heterogeneous' databases), prior redescription is a *necessity*. Unlike the redescription of a regular database, the process of mapping from a corpus to an abstract model determines not only the representation of domain variables but also the sampling of cases. Thus it is more precise to refer to 'redescription' as *abstraction* from corpus to model (figure 4).

Non-regular heterogeneous databases consist of long sequences of complex structure (in our case, texts divided into grammatically annotated text units). They do *not* consist of regular, easily sampled, independent cases. A key aspect of *corpus* data redescription is the determination of *case scope and interaction*. Abstraction from a corpus must identify 'cases' in the source by considering aspects that explicitly deal with neighbouring elements: prior or posterior nodes, for example. There are at least two aspects of this interaction.

a) *Subsumption*. For example, noun phrases that are nested in one another. Such cases cannot be said to be strictly independent, due to their co-occurrence in *language production*.
b) *Overlap*. One consequence of (a) is that elements of a case may partially overlap with elements of another case, or variables concerning neighbouring nodes may overlap. The most important example of this is *grammatical interaction* (e.g., two *conjoined clauses*, as in "*I am and I am not*" [ICE-GB: W1B-001 #059]).

We propose an initial method for determining cases for our grammatically oriented investigation, based around Fuzzy Tree Fragments and their foci. (Sections 5.2 and 5.3 describe this process using real examples). We may deal with overlap statistically by first modelling the relative independence of cases. This was not possible in our experiment, but it would be preferable for investigations of less frequent phenomena.

Finally, observe that abstractive redescription is only one part of an ultimate KDC process for linguistic research. We propose the construction of a 'scientific workbench' system supporting a *KDC cycle* (figure 5) consisting of (i) *exploration*, (ii) *abstraction*, (iii) *analysis*, including ML, and (iv) *concretisation* (re-application of results to the corpus). ICECUP permits corpus exploration. The KDC cycle consists of three further stages. *Abstraction* elicits the abstract linguistic model. *Analysis* applies computation to search this hypothesis space for useful and significant hypotheses between variables in the abstract model. These may then be evaluated by domain experts. A cycle encourages experts to critique their own hypotheses about the subject of study through repeated 'experiments' (Kondopoulou, 1997).

We build on the perspective of Shadbolt and Cupit (1996) by stressing the importance of a final step, *concretisation*. This is particularly important in linguistics because of the necessity to draw attention to the highly interpreted nature of the data source. What they term

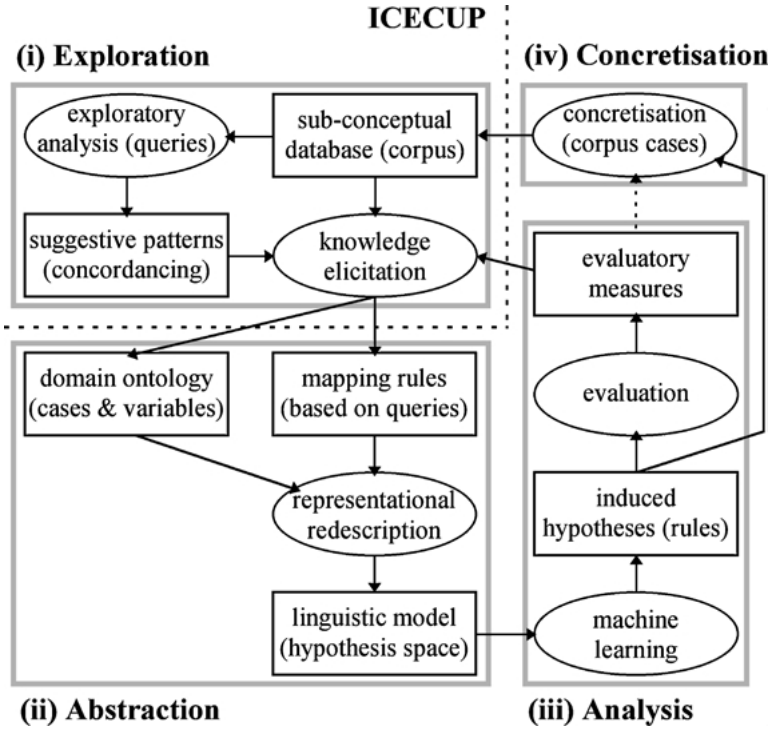


Figure 5. The KDC cycle (after Cupit & Shadbolt, 1996).

the "sub conceptual database"—the linguistic model—must *necessarily* be abstracted from the annotated text in the corpus in order to permit generalisation (figure 4). The annotation should not simply be taken as given, as we commented in Section 4. Moreover, different linguistic traditions impose distinct analyses on the same examples.

Therefore, in order to contrast automatically generated hypotheses with those in the literature, we must aim to identify and extract cases in the corpus that support and contradict these hypotheses. As we shall see, our pilot experiment is limited by the practical difficulty of this final step. In the conclusions we discuss how concretisation could be supported more effectively in an integrated workbench.

5. An experiment in KDC

We proposed the following experiment. We would ask an expert linguist to manually specify a set of variables and mapping rules, and employ a batch process to extract the abstract model from the corpus. Cases in this model would then be processed by the ML algorithm and the results presented to the expert for discussion in the light of current linguistic theory.

Naturally, there would be a number of limits to such an experiment, the most important being the aforementioned issue of weak concretisation.[9] We used ICECUP after the fact to locate examples by applying conjunctions of FTFs, but this is not the same as recovering the specific examples that led to a particular rule being selected. Moreover, one could not easily use the examination of these cases to cue the elicitation of new variables. This meant that the representation could not easily be refined by the KDC cycle proposed in Section 4 and we would therefore not be able to study theory evolution in this context.

A final issue is the sampling method employed. We reduce the prior likelihood of case interaction by employing subsampling. In some low-frequency investigations, this approach would not be feasible, requiring us to exploit the maximum amount of information from our corpus. We therefore chose a reasonably common structure to investigate: *noun phrase postmodification*.

5.1. Postmodification in noun phrases

In this experiment, we focus on the choice between two types of *noun phrase postmodifying* (NPPO) *clauses*, which we exemplify in Table 1. With certain restrictions, a choice is available between these two types of postmodifying clause (Quirk, Greenbaum, Leech, and Svartvik, 1985, p. 1263). More precisely, this choice is between a nonfinite (reduced) relative clause (an *-ing* clause (1) or an *-ed* clause (2)) and a full relative clause in which the relative pronoun (e.g., *who* or *which*) functions as the subject ((1a) and (2a)).

The purpose of this experiment is to examine the factors which influence this choice, using the ICE-GB corpus as our dataset. In other words, we are looking for the rules, if any, that predict the choice of one clause type over the other. ICE-GB yields 5,342 full relative clauses in which the subject is a relative pronoun. This compares with 3,792 postmodifying nonfinite clauses (1,370 *-ing* clauses, 2,422 *-ed* clauses).[10]

We divide the variables into two types, (i) *internal*, that is, variables relating to the postmodifying clause itself, and (ii) *external*, that is, variables relating to the host clause,

Table 1. Two types of clauses that postmodify noun phrases.

Type 1: Nonfinite clause	Type 2: Relative clause
(1) people *living in Hawaii.*	(1a) people *who live in Hawaii*
(2) the book *published in London*	(2a) the book *which was published in London*

i.e., the clause containing the NP. We hypothesize that among the internal variables, the transitivity of the verb will be an important factor (see Table 2). In particular, if the verb is copular, then the postmodifying clause will most likely be relative. This is because the most common copular verb, *be*, can not normally occur in postmodifying nonfinite clauses:

the man who is the president $\sim^*$ the man being the president

However, with other, less common copular verbs, the choice is available:

the athlete who is lying second $\sim$ the athlete lying second

The external variables have been selected to describe the syntactic environments in which the postmodified NPs occur. Among the external variables, we expect that the function of the NP will be significant. In particular, if the NP is the subject of the host clause, we would expect to find a nonfinite postmodifying clause, rather than a relative clause, when a choice is available. Subject NPs tend to be less complex than NPs in other positions (De Haan, 1986). Since nonfinite postmodifiers are more compact than their relative counterparts, they would be the obvious choice when they occur in subject NPs.

All the variables considered in this first pass are summarised in Table 2.

5.2. Declaring domain variables

In order to extract examples of NPPO clauses, we employ Fuzzy Tree Fragments. First, we use FTFs to determine the set of matching cases. Second, we define a set of relevant discrete variables and employ FTFs to specify the instantiation of these variables.

UNIT permits discrete variables to include *acyclic hierarchical laddering*, i.e., typological hierarchies where subvalues are mutually exclusive. Thus, in figure 6, the variable 'VP transitivity' simply picks up the transitivity feature in the verb phrase in the postmodifying clause. If there is no verb phrase in this position, the variable will be undefined. If there is more than one (probably an error), then the last match is taken.

In the 'form' variable, in order to determine 'relative form NPPOs' the identifying 'relative' feature must be specified in a subject NP that has a *pronoun* head. The representation does not rule out the 'ed participle' or 'ing participle' features. It would be an error if both cases matched.

These FTFs are used to collect information from within and below the focus node—what we might call 'internal' variables because they are internal to the clause under investigation. It must also be possible to use FTFs to obtain 'external' information from siblings and

Table 2. Variables and values specifying the domain model (in the first pass).

Variable	Value	Explanation
Form	Relative, nonfinite	
VP transitivity	intransitive (intr) copular (cop) *transitive*: monotransitive (montr) ditransitive (ditr) dimonotransitive (dimontr) complex transitive(cxtr) trans (trans)	intransitive = no verb complement (e.g., David *spoke*) copular = subject complement present (David *is* ill) monotransitive = only direct object present (We met David) ditransitive = indirect object and direct object present (We *told* David the news) dimonotransitive = only ind. object present (We *told* David) complex transitive = direct object and object complement present (It *made* David angry) trans = complement is direct object plus nonfinite clause (We *asked* David to leave)
Subordination	complex, simple	complex = clause contains a subord. clause simple = contains no subordination
Verb	be, other	verb is any form of *be*, or other verb
Other NPPO?	yes, no	presence or absence of any other NP postmodifier
NP function	subject, prepositional complement, subject complement, *object* (direct, indirect), adverbial, other	The function of the NP in the host clause
Constituent between	present, absent, NPPO	presence or absence of any constituent between the NP head and the NPPO
Host clause form	finite, nonfinite, other	finite = main verb has tense (past/present) nonfinite = main verb has no tense
Host clause VP transitivity	intransitive, copular, *transitive* (mono-, di-, dimono-, complex, trans.)	intransitive = verb complement absent transitive = verb complement present (see above) copular = subject complement present
NPHD number	singular, plural	NP head is singular or plural
NPHD realisation	noun, pronoun, nominal adjective, numeral, proform	NP head is noun, pronoun, etc.
NPPRs?	yes, no	presence or absence of any premodifier
Text category	spoken (..) written (..)	sampling variable, see figure 1

above this node. For example, we include variables to determine if there are any other postmodifying clauses, any premodifying clauses in the NP, the realisation of the noun phrase head (e.g., noun, pronoun, etc.) and its number (figure 7).

Most of our FTFs use fixed, rather than 'fuzzy', relationships between elements. However, sometimes it is necessary to specify nodes that cannot be located deterministically. For example, in order to determine whether another NPPO clause is present in the same NP, we specify that it could occur *before or after* the focus NPPO clause. A more significant problem occurs if not only do we wish to detect the presence of a node at some unspecified distance but *form a decision* on the basis of that node.

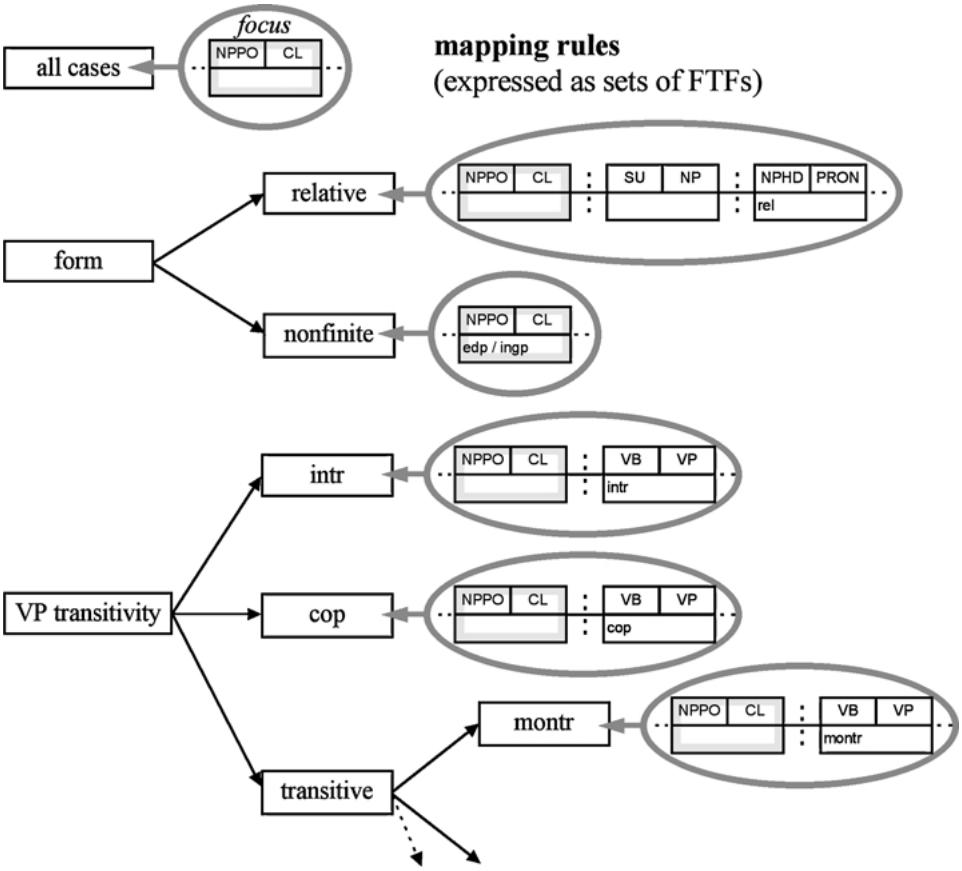


Figure 6. Defining a case, and simple domain variables with mapping rules.

We wish to determine the form of the 'host clause' of the NPPO clause, and the transitivity of the verb phrase within it. The host clause is defined as the *closest ancestor* clause. However, FTFs do not contain a 'closest ancestor' link, because this contradicts the notion of a declarative representation. Even if they did, we would have to ensure a strict ordering on evaluation, in other words, guarantee that we first, identify the nearest host clause, and second, determine nodes relative to it.

However, although FTFs are declarative, the mapping process itself is procedural. Figure 7 illustrates the principle. We declare an FTF for the domain variable. This *defines the scope* of the variable. This FTF employs an 'ancestor' relation between the case node and the host clause. (We ensure that the closest clause is chosen in preference to the furthest). We also include a *secondary focus point*, causing the focus to 'shift' to another node in the corpus tree. The mapping rules for the values below this point are then evaluated against this node. This guarantees a single unambiguous decision based on the closest node to the NPPO clause.

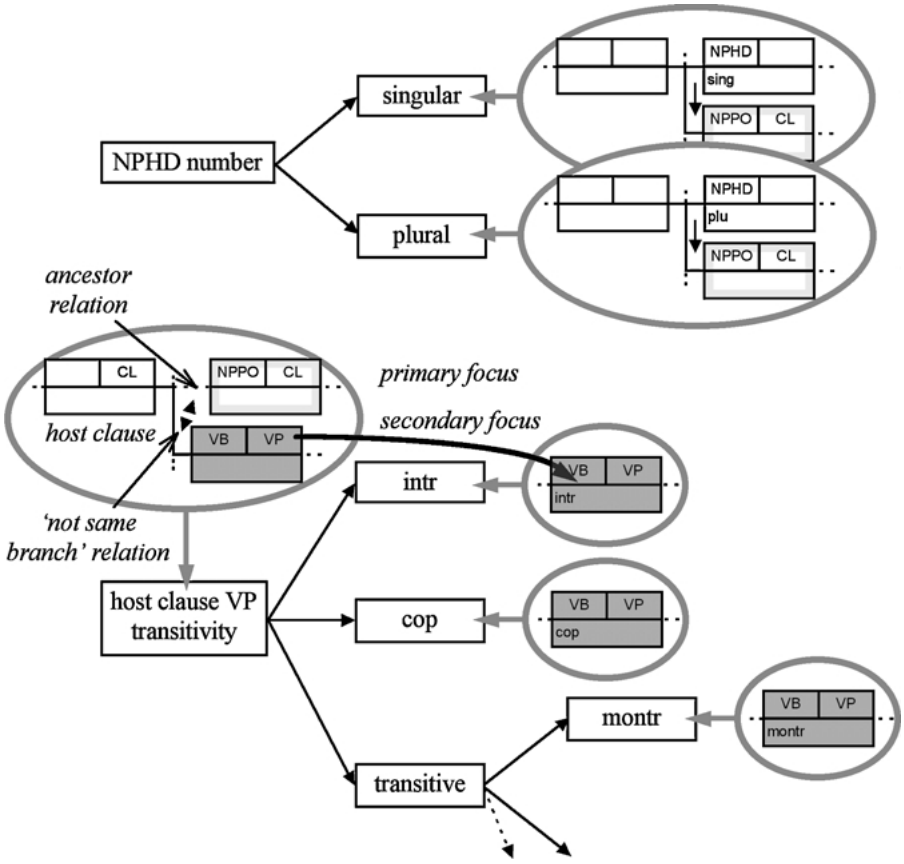


Figure 7. Defining complex external variables.

We define 13 initial variables (Table 2), including the sociolinguistic 'text category' hierarchical variable. ICE-GB uses the TOSCA/ICE analysis of transitivity (Greenbaum, 1996a), which distinguishes between a variety of subtypes (monotransitive, ditransitive, dimonotransitive, etc.). There is no unanimous agreement in the literature about the most appropriate level of description. As UNIT can exploit a hierarchical variable typology, we opt to introduce a broader 'transitive' group which brackets subtypes together, which we distinguish from intransitive and copular types. We group direct and indirect objects similarly.

5.3. Redescription in action: abstracting the domain model

We generate our domain model by applying these FTF mapping rules to the corpus. Each relative or nonfinite NPPO case is identified, and the set is subsampled by a random criterion.

Each case in the corpus generates a corresponding case in the abstract model, expressed as a tuple of attribute-value pairs. A decision procedure for each variable is applied in

turn. This determines the value of the variable by hierarchical descent, testing to see if the mapping rule (disjunctive sets of FTFs) for each value matches the tree *at the same point* as the case focus.

Figure 8 demonstrates this process. FTFs for VP transitivity are applied to the same tree as the FTF for 'form = nonfinite'. We determine that the VP in question is monotransitive. We then proceed to the next variable, 'subordination'. If the postmodifying clause contains another clause, it is complex. Here, the absence of a match determines that it is marked as 'simple'. Figure 8 illustrates other relevant nodes and information that may be extracted.

If no match is found for an FTF under a variable then it is left undefined. The last variable is the sociolinguistic 'text category', where the value is simply copied from the corpus. The result is a large regular database table, in the first pass consisting of 4,579 cases from a 50% sampling rate.

5.4. Analysing the domain model

The UNIT machine learning algorithm was used to explore this dataset, searching for rules predicting the variable 'form'. UNIT (*University of Nottingham Induction Tool*: Wallis, 1993; 1999) is a general-to-specific *similarity-based learning* (SBL) algorithm similar to

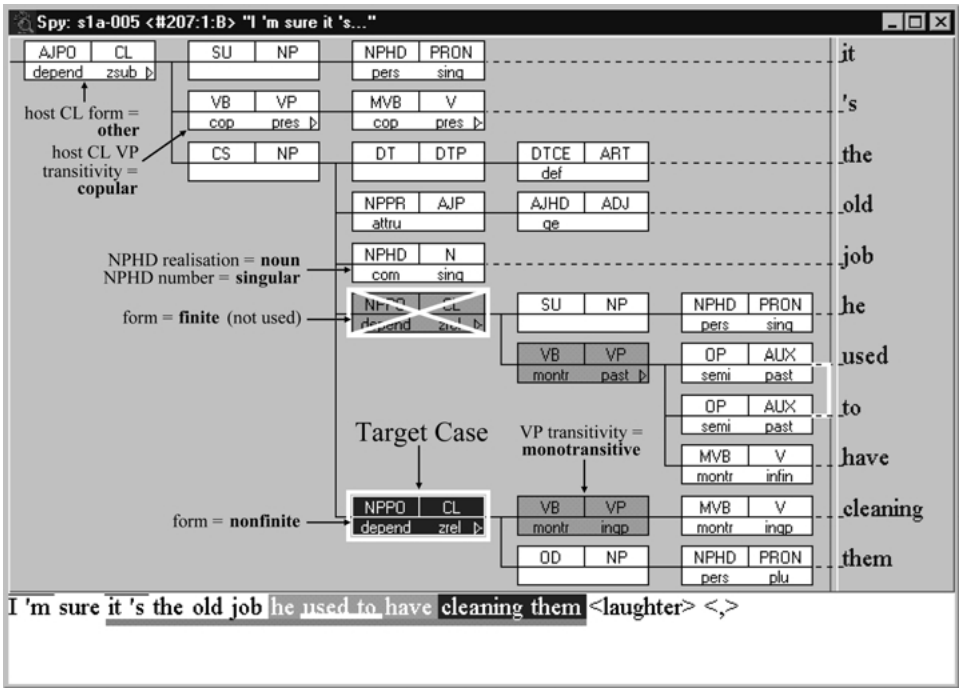


Figure 8. Using FTFs to identify a case and extract its variables. FTFs for '**form = nonfinite**' and '**VP transitivity = monotransitive**' have been applied.

CN2 (Clark and Niblett, 1989), which employs a top-down 'beam search' that can be pruned using a variety of methods.

For our purposes, UNIT has a number of features that are useful for knowledge discovery. These include: a flexible maximisation function, options for exhaustive parallel search and statistically sound search, and the exploitation of given hierarchical structure in target and source variables. UNIT generates production rules, either as a strictly independent 'unordered' rule set or as an ordered 'decision list'.

Each rule is evaluated according to a maximisation function, *utility*, and tested for significance according to its distribution over the target concept. For example, the following rule (from Appendix 2) has 174 supporting and 61 counter examples. It therefore has a *Laplace accuracy* (Niblett and Bratko, 1987) of 74%. However, it has a *coverage* of only 5% (174 + 61 out of the 4,434 cases where 'form' was defined).

independent rule for FORM	**+ve**	**−ve**	**cov**	**fit**	**acc**	**utility**
IF NP_FUNCTION = SUBJECT_COMP **AND** NPPRS = NO **THEN** FORM = RELATIVE	174	61	0.053	0.066	0.737	0.569

In this experiment, *utility* is mostly based on Laplace rule *accuracy*, but with smaller components of *coverage* (the applicability of the rule) and *fitness* (the accuracy of the inverse). Formally, we use UNIT to search for rules that hold

$$utility = coverage^{0.07} \times fitness^{0.03} \times accuracy_L^{0.90} > 0.5.$$

This function permits the tradeoff of these different measures against one another, so a more general rule with medium accuracy may be judged to be more salient than a rule with higher accuracy but of very low coverage. The exponents bias *utility* so that, for example, accuracy is prioritised over coverage.

UNIT employs a hierarchical generalisation of the log-likelihood statistic (Haberman, 1978), which is χ^2 distributed, to test rules for significance. This determines if the conditional part of the rule (the left hand side of the '**THEN**') significantly alters the distribution over the variable we are attempting to investigate (in this case, 'form'). We test for significance (a) to determine the statistical viability of a single rule, and (b) to determine if each successive refinement of the rule significantly affects the consequent. Thus, for example, the discovery of the rule above tells us (i) the rule itself is significant (in terms of significantly altering the distribution of cases over 'form' at a $p > 0.95$ level), (ii) a simpler rule consisting of only *one* of the preconditions is significant at the same level, and (iii) the rule above is significantly different from this simpler rule.

Employing significance in this way is more strict than is typical in machine learning.[11] UNIT also searches the space of possible rules differently. *Decision tree* algorithms (e.g., C4.5; Quinlan, 1993) employ a 'divide and cover' strategy that constructs a tree by repeated

partitioning of the target concept. The choice of initial partition and condition determines the context of future tree elaboration. Hence, these structures are often difficult to interpret (Michie, 1986), even when paths are converted into rules.[12] On the other hand, *decision list*, or 'ordered rule', algorithms (e.g., CN2; Clark and Niblett, 1989) attempt to find a single 'best' rule predicting a target concept and then partition the target according to those covered, again, performing a shift of origin. These structures also tend to be opaque (see Section 6.4).

UNIT offers an *unordered rule set* algorithm (Wallis, 1999b) that searches for all rules in parallel across the entire training set. The resulting rule set is thoroughly 'decomposable', i.e., a rule can be considered in isolation from all others. An apparent disadvantage to this approach is the presence of a large amount of 'redundancy' in the rule set. This means that there may be many different rules covering the same set of cases (and entailing the same result). Note that such redundancy can be very useful in knowledge discovery, however: two syntactically dissimilar rules correlating with a particular phenomenon are putative *alternative explanations*. It is up to researchers to interpret these results and defend their interpretations.

This class of SBL algorithms generates candidates through incremental specialisation, that is, by taking a candidate rule and adding new conjunctive preconditions or refining existing ones. Rules that fulfill certain criteria (e.g., an increasing maximisation function) survive to the next round where they are specialised again. Since the set of candidates at any stage can grow exponentially, it is common to employ *competitive pruning* (i.e., any method where the presence of one rule candidate excludes another, such as, a limited number of candidates).

In our experiment, we prune the search space by the requirement (a) that the *utility* function must increase and (b) that each specialisation step must significantly alter the distribution over the consequent. This 'semi-exhaustive' search produces a rule set where rules are not merely unordered, but where the *absence* of an expected rule can be highly informative. We suspect that the latter could be particularly useful in driving further evolution of the domain model.

6. Results and discussion

6.1. Linguistic analysis

We applied UNIT to identify independent rules predicting the primary variable, 'form', from the other domain variables, obtaining the rule list in Appendix 1. We initially applied two kinds of search: a standard one that 'evolved' hierarchical preconditions over each pass and another that exhaustively searched these variables. This allows us to avoid the implicit danger in 'laddering up' in preconditions, namely that superordinates in a putative hierarchy cause the search to be pruned prematurely (superordinate values must significantly affect the distribution of cases across the target concept before any subordinates are considered and a subordinate child must significantly differ from its parent).

For convenience, we divided the rules according to the domain variable type. The first set of rules relates to the transitivity of the postmodifying clause.

independent rules for FORM	+**ve**	−**ve**	**cov**	**fit**	**acc**	**utility**
IF VP_TRANSITIVITY = COP **THEN** FORM = RELATIVE	838	28	0.189	0.314	0.967	0.832
IF VP_TRANSITIVITY = DIMONTR **THEN** FORM = RELATIVE	183	9	0.042	0.069	0.948	0.703
IF VP_TRANSITIVITY = TRANS **THEN** FORM = RELATIVE	250	27	0.060	0.094	0.900	0.694
IF VP_TRANSITIVITY = DITR **THEN** FORM = RELATIVE	211	18	0.050	0.079	0.918	0.694
IF VP_TRANSITIVITY = INTR **THEN** FORM = RELATIVE	660	302	0.210	0.247	0.686	0.612

As we predicted in Section 5.1, the results indicate an interaction between transitivity and the form of the postmodifying clause. The first rule predicts a strong correlation between a copular verb and a relative clause, or, conversely, that nonfinite postmodifying clauses are unlikely to contain a copular verb. In fact, the corpus yields only 41 instances of a postmodifying clause in which the verb is copular. Many of these are fixed expressions, as in (a) and (b), or are otherwise syntactically distinctive (c). The codes refer to their location in ICE-GB.

a) For some this so-called age of plunder is a dream *come true*$_{[S2B\text{-}023\,\#003]}$.
b) So we won't use that for the time *being*$_{[S1A\text{-}045\,\#176]}$.
c) Anyway enough *being poetic*$_{[W1B\text{-}006\,\#022]}$.

With the possible exception of (a), it would be difficult to think of a corresponding relative clause, so, in effect, no choice is available between our two postmodifying clause types. The first rule correctly predicts the fact that nonfinite postmodifying clauses with a copular verb are rare.

The other rules relating transitivity to clause type are less strong, and more difficult to interpret, not least because the transitivity values are mutually exclusive. The second set of rules relates text category to clause type, as shown below:

independent rules for FORM	+**ve**	−**ve**	**cov**	**fit**	**acc**	**utility**
IF TEXT_CATEGORY = "DIALOGUE" **THEN** FORM = RELATIVE	721	253	0.213	0.270	0.740	0.657
IF TEXT_CATEGORY = "SPOKEN" **THEN** FORM = RELATIVE	1551	771	0.507	0.581	0.668	0.652
IF TEXT_CATEGORY = "PUBLIC" **THEN** FORM = RELATIVE	436	156	0.129	0.163	0.736	0.622
IF TEXT_CATEGORY = "DIRECT CONVERSATIONS" **THEN** FORM = RELATIVE	262	85	0.076	0.098	0.754	0.603
IF TEXT_CATEGORY = "PRIVATE" **THEN** FORM = RELATIVE	285	97	0.083	0.107	0.745	0.602

The text categories are arranged hierarchically, and the rules reflect this to some extent. The hierarchy of relevant text categories is shown at the left of figure 1. The most general rule predicts a correlation between relative clauses and spoken texts, while the following rule predicts the converse:

independent rule for FORM	**+ve**	**−ve**	**cov**	**fit**	**acc**	**utility**
IF TEXT_CATEGORY = "WRITTEN" **THEN** FORM = NONFINITE	1139	1118	0.493	0.596	0.505	0.507

Taken together, these two groups of rules constitute the clearest linguistic results of this experiment. Briefly, they predict that relative clauses will typically occur in speech, and that nonfinite modifying clauses will typically occur in writing. This confirms earlier findings by Biber (1988), who found that postmodifying nonfinite clauses are much more common in writing than in speech. Using the *London–Oslo–Bergen* (LOB) corpus as his dataset, Biber found 4.3 instances per 1,000 words in writing, compared with 1.9 per 1,000 words in speech.[13] In ICE-GB the corresponding figures are 4.3 in writing and 2.4 in speech.

The other text category rules serve to refine this general result, namely, that relative clauses will more often be selected in speech. They indicate that while this is generally true, it is more precisely in dialogues—both public and private—that this is the case. Interestingly, no rule is produced to predict the form in monologues. This suggests that the significant contrast is between *dialogue* and writing, not between *speech* and writing. Among the dialogues, only the category of *direct conversations* generates a rule predicting form. This may be explained by the preponderence of conversations in the corpus: 90 of the 300 spoken texts are conversations.

Very few rules predict the nonfinite option. In the broadest terms, this means that nonfinite postmodifiers are difficult to predict. They are relatively rare, and apart from saying that they typically occur in writing, the rules do not strongly predict any other major factor. This suggests that other variables need to be examined, though it is likely that these are ones which have not been encoded in the corpus. That is to say, the major influences affecting the choice of a nonfinite postmodifier may not be syntactic at all. Other factors, such as the formality of the texts, and their information content, may also be contributing factors.

6.2. Developing the analysis

Since we discovered that 'all transitives' was not a good predictor for form, we decided to remove this intermediate node in both transitivity variables (VP and host clause VP). This ensures that leaf values are considered in the evaluation, and removes any advantage of exhaustively searching these variables.

We observed that transitivity in general does not have a major effect on form, although subtypes of transitivity do. ICE-GB follows Quirk, *et al.* (1985) in explicitly representing subtypes of 'transitive', although many corpora (such as the BNC) do not.

Following some discussion, we proposed to ladder the 'form' variable down under 'nonfinite' (figure 9), obtaining a simple hierarchical variable with two new leaves, '*ing* participle'

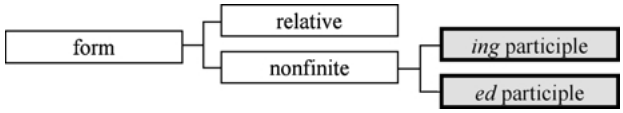


Figure 9. Laddering down on the target concept.

and '*ed* participle'. (Laddering down is feasible if the domain model may be resampled). We also introduced a number of additional variables, such as 'speaker age'. After resampling and reanalysis, we obtained the unordered rules listed in Appendix 2.

6.3. Further linguistic analysis

If we return to the question of transitivity (Section 6.1 above), it is interesting that all the transitivity types generate a rule predicting a relative clause, with the exception of monotransitive and complex transitive. This is not surprising when we consider that over 64% of the nonfinite clauses are *-ed* clauses. Clauses of this type are passive, and the passive voice precludes a copular or an intransitive verb. Returning to the corpus data, we observe that 85% of *-ed* clauses have a monotransitive verb, while 13% have a complex transitive verb. These transitivity types, therefore, are predictors of a nonfinite clause, but only if we distinguish between the two subtypes, *-ed* clauses and *-ing* clauses. When we make this distinction, we obtain the following rule.

independent rule for FORM	+**ve**	−**ve**	**cov**	**fit**	**acc**	**utility**
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = INTR **THEN** FORM = INGP	281	8	0.065	0.436	0.971	0.933

This rule correctly predicts that an intransitive verb indicates an *-ing* clause, rather than an *-ed* clause. As we would expect, we also obtain a rule which states the converse of this:

indpendent rule for FORM	+**ve**	−**ve**	**cov**	**fit**	**acc**	**utility**
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = MONTR **THEN** FROM = EDP	1039	333	0.307	0.858	0.757	0.710

The following rule is a refinement of this one, with a higher degree of accuracy:

indpendent rule for FORM	+**ve**	−**ve**	**cov**	**fit**	**acc**	**utility**
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = MONTR **AND** SUBORDINATION = SIMPLE **THEN** FORM = EDP	982	285	0.283	0.811	0.775	0.719

According to this rule, the correlation between a monotransitive verb and an *-ed* clause is even stronger if the clause contains no subordination ('subordination = simple'). Returning to the corpus data, it becomes clear that many of the *-ed* clauses are quite short and simple,

often consisting of a monotransitive verb alone ("the university *concerned*", "the people *involved*"), or of a monotransitive verb followed by a *by*-agent phrase ("the role *played by middle management*"). In all of these cases, the *-ed* clause contains no subordination. This is in keeping with nonfinite postmodifying clauses generally, which are more compact and shorter than their relative clause equivalents. While this is common knowledge in grammar, the rule above adds a very useful refinement, by stating that the most compact of all the nonfinite postmodifiers is the *-ed* type, and specifically, those containing a monotransitive verb.

A number of other new rules were found which involved some of the new variables (Appendix 2). However, these were difficult to explain without more effective concretisation.

6.4. Decision lists: comments

To compare the effectiveness of ordered and unordered sets of rules in knowledge discovery we also presented our domain expert with sequences of ordered rules obtained by UNIT using a cover-and-remove method. The following list was obtained from the first pass.

ordered rules for FORM	**+ve**	**−ve**	**cov**	**fit**	**acc**	**utility**
IF VP_TRANSITIVITY = COP **THEN** FORM = RELATIVE **ELSE**	838	28	0.189	0.314	0.967	0.861
IF VP_TRANSITIVITY = INTR **AND** SUBORDINATION = COMPLEX **THEN** FORM = RELATIVE	54	5	0.016	0.029	0.902	0.665
ELSE						
IF VP_TRANSITIVITY = TRANSITIVE **AND** SUBORDINATION = SIMPLE AND TEXT_CATEGORY = "WRITTEN" **THEN** FORM = NONFINITE	918	516	0.392	0.489	0.640	0.619
ELSE						
IF SUBORDINATION = COMPLEX **AND** OTHER_NPPO = YES **THEN** FORM = RELATIVE	51	12	0.028	0.040	0.800	0.619

However, while our expert found that he could justify the first one or two rules (via the unordered rules, where they also appeared), by the third rule, explanation became more difficult. The decision list structure could not easily be decomposed into separate hypotheses, either in order to test it against the corpus or to compare it with claims in the literature.

Secondly the structure specifically excludes alternative explanations for the variance illustrated. Our expert markedly preferred independent rules and was able to apply them in order to revise his experimental design.

7. Conclusions

7.1. Summary

We have demonstrated the applicability of KDD to corpus linguistics. In addition, we have demonstrated that the question of *representation* is central. The data source is so

complex, and the possible range of interactions so rich, that it is essential to first identify and abstract potentially salient features in the form of a domain model. Our approach differs from 'learning' within natural language processing (Briscoe, 1996; Fang, 1996) by emphasising that meaningful results are to be obtained at a higher level of abstraction than the corpus annotation itself. We share with the 'Learning Language as Logic' (LLL) proposals of Muggleton (1998) the recognition that significant additional background knowledge is necessary in order to obtain linguistically meaningful results. We also agree that research should be *goal directed*, toward the solution of particular linguistic questions (e.g., past tense in verbs, Mooney and Califf, 1995).

Where we primarily differ from Muggleton, *et al.* is in our perspective with respect to the imperfection and partiality of our own background knowledge of language, and the limitations of existing annotated corpora. The argument underlying the '3A perspective' introduced in Section 4 is that it is not possible to make generalised statements about naturally occurring samples of language unless one first collects such samples and formalises the process of annotation and abstraction. Since this is necessarily a process requiring linguistic intervention, as a starting point our representations must be commensurable with current linguistic knowledge. The KDD bottleneck, therefore, is a KA bottleneck.

Nor should we be tempted to overgeneralise from results obtained from highly linguistically restricted samples such as *database queries* (Mooney, 1997; Muggleton, 1998), *computer manuals* (Black, Lafferty, and Roukos, 1992) or *dictionary examples* (Fang, 1996). To put it mildly, *unrestricted natural language is noisy*. We would expect 'crisp' accurate rules only in two situations. Either the corpus annotation was deterministically introduced and our results reflected this, or we are mistakenly representing the same domain relationship twice (e.g., relating the number of objects in a VP to the transitivity feature).

While LLL methods have had some good results on restricted domains, no amount of logical power can compensate for an inadequate sample or an inadequate representation. Moreover, if linguists cannot explain, interpret and justify networks of logical statements, it is difficult to see the benefit of learning. On the other hand, linguists require ever more powerful tools to apply their own knowledge to corpora. A workbench that they can use unaided is therefore necessary.

Our proposals for the identification of cases and the instantiation of variables are, for the most part, linguistically intuitive. Fuzzy Tree Fragments are becoming established within corpus linguistics, and their declarative simplicity and familiarity appeals to linguists. Employing them in mapping rules to translate from the corpus to an abstract model is entirely consistent with their current research usage. The main difference here, as compared to their use in an exploration tool such as ICECUP, is in the degree of necessary *formalisation* in the abstraction process, requiring the expert specification of explicit variables and mapping rules. We would aim to ease this KA bottleneck by a number of techniques, including *laddering* and *structured interviewing* (Corbridge *et al*., 1994). We may also exploit the corpus representation to permit the system to suggest ways in which a variable may be completed. The sampling process can also help identify if a variable is incompletely instantiated (for example, if a domain variable doesn't apply to a significant number of cases).

We discovered one problem with employing FTFs for abstraction. FTFs are declarative, model-based representations, but we found it necessary to exploit a procedural approach in order to abstract features of the *host clause*. Future work should include support for making any necessary procedural knowledge explicit.

7.2. Data analysis with UNIT

Once a domain model was constructed, we were able to explore the viability of the UNIT ML algorithm for data analysis in this domain.

UNIT (Wallis, 1999b) permits the *comparative evaluation* of many simple hypotheses in the form of inference rules. As discussed in Section 5.4, UNIT can be used to obtain a list of all rules that score higher than a particular threshold (ordered by the score), containing a set of preconditions, each of which significantly contribute to the predictive power of the rule. Moreover, we can exhaustively search a significance-limited subset of all possible hypotheses within the domain model. The algorithm explores the space by starting with a very general hypothesis and then proceeds to look for more specific ones.

There is thus both a positive, confirmational aspect of ML employed in this way ('discovery'), and a negative, disconfirmational aspect. We can show that other possible explanations considered either do not apply, or are not as accurate predictors. This does not mean that no explanations *exist* that subsume discovered hypotheses. However, it does mean that *no explanation considered in the abstract model* suffices as effectively.

There are a number of possible reasons why hypotheses in the literature may not be reflected in results. Variables that might have proved more predictive may be absent. Defined variables may be limited by the inadequacies of the representation (either poor annotation or poor translation from the annotation to the domain model). The corpus may be an unrepresentative sample or cases may be too infrequent. Or, the literature may be inaccurate and unrepresentative. The crucial point to note is that in order to address this question, an investigator must re-examine the corpus, using the results as a guide. In other words, adequate support for concretisation is essential. We discuss the implications of this in Section 7.4.

Finally, we were also able to exploit hierarchical target concepts and *refinement rules*[14] to find rules that separately predicted the *-ing* and *-ed* participle subtypes of nonfinite clauses.

7.3. Exploiting hierarchical variables and unordered rules in discovery

The use of hierarchical variables, although initially unfamiliar, made sense to our domain expert for cases such as *transitivity*, once a linguistic motivation was suggested. Their use permitted us to experimentally evaluate, in the course of a research investigation, the predictive value of levels of description of transitivity (an issue of debate in grammar). In fact, we found that it was useful to discriminate between subclasses of transitivity in this case, and therefore, we removed the laddering.

A second application came with the hierarchical refinement of the target concept (figure 9). We were able to find a cluster of cases containing a systematic set of attributes predicting *-ing* and *-ed* clauses within the nonfinite cases. Although these were represented as

refinement rules, this caused no particular problem of interpretation. We can therefore state with some confidence that such hierarchical variables are indeed valuable and meaningful within knowledge discovery.

Our expert was, however, much more marked in his preference for employing unordered rules rather than decision lists. Decision lists presented no new information (clusters in this domain appear to be relatively separable anyway) but were harder to interpret. The linguist had to convert them into unordered rules first, which meant that many later rules were simply discarded due to excessive complexity. Second, it was not possible to view alternative explanations for the phenomena. Even if an exhaustive *search* was applied (subject to significant improvement, say), lesser-scoring explanations were not available for comparison.

7.4. Future work

We employed a '3A' perspective that concentrates on representational meaning. Provided that the domain model was abstracted using linguistically meaningful terms, rules generated should be readily understandable. Here, the simplicity of production rules are a great benefit. However, understanding what the terms employed in a rule mean is not the same as understanding *why* the rule is supported. The principal limitation of our experiment was that having obtained a rule it was difficult to *concretise* it by directly extracting both supporting or counter examples for a particular rule, either at the abstract (domain model) or corpus level. This had two consequences.

The immediate result was that the linguist could only obtain similar examples, imperfectly, from the corpus using ICECUP. ICECUP does not use FTF foci to specify a case in the way we described. Due to subsampling, recovery could not guarantee the identification of specific cases. With more complex rules (e.g., with multiple preconditions) this is exacerbated.

The second consequence derives from the first: if concretisation is difficult, then it becomes harder to use the results to cue further acquisition. This is a method exploited by *repertory grid tools* (Shaw and Gaines, 1987), for example, which use the results of analysis to focus elicitation on undiscriminated clusters where new variables may be necessary. In our case, poorly covered or unsatisfactorily explained cases could be used to propose new FTFs and variables, or refine existing ones. Arguably, ICECUP already employs a weakened form of this notion in its cyclic exploratory procedure (Wallis, Nelson, and Aarts, 1999).

In summary, these limitations support the argument that a workbench is required to integrate and control the overall process, and in particular, support concretisation.

We envisage an integrated approach to the visualisation of abstracted cases in the domain model, in other words, so that each distinct example in the corpus (a sentence plus the focus node locating the example within the sentence), can be viewed side-by-side. Second, all cases covered by a rule, whether supporting or contradicting, should be accessible from the rule. (Either the ML algorithm should pass this information back to the workbench or the workbench should obtain it by re-applying rules to the linguistic model.) Third, there should be methods for comparing how rules overlap with respect to these cases. This is to allow the researcher to contrast competing hypotheses covering some or all of the same ground. At

present, the closest we can get to this is through 'coverage maps' (see Figures 10 and 11). Finally, we believe that it would also be beneficial to evaluate expert-specified hypotheses against the corpus to suggest explanations of why specified rules were not found by the algorithm.

However, it is one thing to parameterise the architecture in this way, but any system must, be usable by linguists unaided (like ICECUP). There is a real research question here.

Other weaknesses identified include the inability to test hypotheses directly (e.g., explicitly reproduce those in the literature and test them) and the lack of computational support for the acquisition process *from* the corpus. Our sampling method was also potentially problematic, as we could not address the question of case interaction. All of these remain areas of future research.

Despite this, our experiment clearly *does* support the applicability of knowledge discovery techniques to corpus linguistics and, more specifically, the viability of our approach. We have demonstrated the feasibility of knowledge-supported abstraction from annotated corpora, permitting corpus analysis to be exploited in a directed fashion. Our ML component exploited this abstracted database to make unexpected discoveries in linguistics. Our results further indicate that a similar approach may be practicable in the analysis of other structured non-homogeneous databases.

Appendix 1: Results of the first KDC pass

Table 3 summarises the rules generated in the first pass.

UNIT can also visualise rule distributions in the form of a 'coverage map' (figure 10). *Coverage maps* (Wallis, 1999b) depict how each rule covers cases in the training set after the set is subdivided by the target variable, allowing us to compare the coverage of different rules. (You can think of this as a kind of multi-dimensional Venn diagram.) The vertical expresses the spread of cases, whereas each vertical 'strip' corresponds to a rule. The shading then indicates whether a case supports or contradicts the rule. In the first example, all rules predict 'form = relative'.

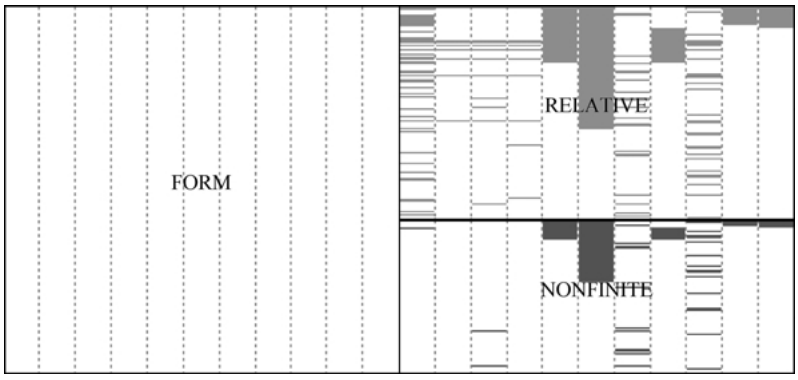


Figure 10. UNIT 'coverage map' of rule distribution for *form*, first section.

Table 3. Rules generated by UNIT, first pass. Significant at the 0.05 level by χ^2. The first section was computed by exhaustively exploring hierarchical variables and holding *utility* $\geq$0.6. The second section lists additional rules that were generated by evolving hierarchical preconditions and requiring *utility* $\geq$0.5.

Independent rules for FORM (first section)	+ve	−ve	cov	fit	acc	utility
IF VP_TRANSITIVITY = COP **THEN** FORM = RELATIVE	838	28	0.189	0.314	0.967	0.832
IF VP_TRANSITIVITY = DIMONTR **THEN** FORM = RELATIVE	183	9	0.042	0.069	0.948	0.703
IF VP_TRANSITIVITY = TRANS **THEN** FORM = RELATIVE	250	27	0.060	0.094	0.900	0.694
IF VP_TRANSITIVITY = DITR **THEN** FORM = RELATIVE	211	18	0.050	0.079	0.918	0.694
IF TEXT_CATEGORY = "DIALOGUE" **THEN** FORM = RELATIVE	721	253	0.213	0.270	0.740	0.657
IF TEXT_CATEGORY = "SPOKEN" **THEN** FORM = RELATIVE	1551	771	0.507	0.581	0.668	0.652
IF SUBORDINATION = COMPLEX **THEN** FORM = RELATIVE	409	137	0.119	0.153	0.748	0.627
IF TEXT_CATEGORY = "PUBLIC" **THEN** FORM = RELATIVE	436	156	0.129	0.163	0.736	0.622
IF VP_TRANSITIVITY = INTR **THEN** FORM = RELATIVE	660	302	0.210	0.247	0.686	0.612
IF TEXT_CATEGORY = "DIRECT CONVERSATIONS" **THEN** FORM = RELATIVE	262	85	0.076	0.098	0.754	0.603
IF TEXT_CATEGORY = "PRIVATE" **THEN** FORM = RELATIVE	285	97	0.083	0.107	0.745	0.602
Independent rules for FORM (second section)						
IF VP_TRANSITIVITY = DIMONTR **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = RELATIVE	80	7	0.019	0.030	0.910	0.619
IF VP_TRANSITIVITY = DITR **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = RELATIVE	93	12	0.023	0.035	0.879	0.611
IF VP_TRANSITIVITY = TRANS **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = RELATIVE	109	18	0.028	0.041	0.853	0.606
IF NPHD_REALISATION = PRONOUN **THEN** FORM = RELATIVE	310	114	0.093	0.116	0.730	0.594
IF NPHD_REALISATION = NOUN **THEN** FORM = RELATIVE	2306	1772	0.891	0.864	0.565	0.592
IF VP_TRANSITIVITY = TRANSITIVE **AND** SUBORDINATION = SIMPLE **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = NONFINITE	925	593	0.332	0.484	0.609	0.579
IF VP_TRANSITIVITY = TRANSITIVE **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = NONFINITE	1001	721	0.376	0.524	0.581	0.562
IF NP_FUNCTION = OBJECT **THEN** FORM = RELATIVE	441	257	0.152	0.165	0.631	0.547
IF VP_TRANSITIVITY = TRANSITIVE **THEN** FORM = NONFINITE	1598	1537	0.685	0.837	0.510	0.529
IF HOST_CL_VP_TRANS = INTR **THEN** FORM = RELATIVE	689	558	0.272	0.258	0.552	0.513
IF TEXT_CATEGORY = "WRITTEN" **THEN** FORM = NONFINITE	1139	1118	0.493	0.596	0.505	0.507
IF NP_FUNCTION = DETACHED **THEN** FORM = RELATIVE	165	89	0.055	0.062	0.648	0.505

Appendix 2: Results of the second KDC pass

On the second pass, the target concept is hierarchically structured (see figure 9), so refinement rules are permitted. This permits the analysis and discovery of variation within the subdomain of *nonfinite* types and, moreover, the comparison of these results with those for the general case. Rules obtained by UNIT are givenin Table 4.

Figure 11 illustrates the comparative distribution of these rules. This is a more complex example because the target is now hierarchical. Rules predicting the new nonfinite subvalues are shown in the lower right of the figure.

Table 4. Rules from second pass induction. Upper part: utility $u > 0.6$, lower: $u > 0.5$. The rules cover 88.9% of 'form' with a mean 76.1% accuracy (total computation: 1,767 rule evaluations).

Independent rules for FORM	+ve	−ve	cov	fit	acc	utility
IF VP_TRANSITIVITY = COP **THEN** FORM = RELATIVE	763	25	0.176	0.292	0.967	0.826
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = INTR **THEN** FORM = INGP	281	8	0.065	0.436	0.971	0.777
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = MONTR **AND** SUBORDINATION = SIMPLE **THEN** FORM = EDP	982	285	0.283	0.811	0.775	0.719
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = MONTR **THEN** FORM = EDP	1039	333	0.307	0.858	0.757	0.710
IF VP_TRANSITIVITY = DIMONTR **THEN** FORM = RELATIVE	179	8	0.042	0.068	0.950	0.704
IF VP_TRANSITIVITY = TRANS **AND** SUBORDINATION = SIMPLE **THEN** FORM = RELATIVE	173	8	0.040	0.066	0.949	0.701
IF VP_TRANSITIVITY = DITR **THEN** FORM = RELATIVE	200	15	0.048	0.076	0.924	0.696
IF FORM = NONFINITE **AND** VP_TRANSITIVITY = CXTR **THEN** FORM = EDP	156	12	0.038	0.129	0.927	0.694
IF VP_TRANSITIVITY = TRANS **THEN** FORM = RELATIVE	247	27	0.061	0.094	0.897	0.693
IF TEXT_CATEGORY = "SPOKEN" **THEN** FORM = RELATIVE	1492	750	0.501	0.570	0.665	0.649
IF FORM = NONFINITE **AND** OTHER_NPPO = NO **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = EDP	692	286	0.219	0.571	0.708	0.644
IF FORM = NONFINITE **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = EDP	771	335	0.247	0.637	0.697	0.644
IF SUBORDINATION = COMPLEX **THEN** FORM = RELATIVE	424	140	0.126	0.162	0.750	0.632
IF VP_TRANSITIVITY = INTR **THEN** FORM = RELATIVE	672	289	0.215	0.257	0.699	0.624
IF NP_FUNCTION = SUBJECT_COMP **AND** NPHD_REALISATION = PRONOUN **AND** NPPRS = NO **THEN** FORM = RELATIVE	42	3	0.010	0.016	0.915	0.590
IF SUBORDINATION = SIMPLE **THEN** FORM = RELATIVE	2193	1716	0.874	0.838	0.561	0.586
IF NPHD_REALISATION = PRONOUN **THEN** FORM = RELATIVE	290	114	0.090	0.111	0.716	0.585
IF NP_FUNCTION = SUBJECT_COMP **AND** NPPRS = NO **THEN** FORM = RELATIVE	174	61	0.053	0.066	0.737	0.569
IF SPEAKER_AGE = "46–65" **THEN** FORM = RELATIVE	1456	1131	0.578	0.556	0.563	0.564
IF VP_TRANSITIVITY = MONTR **AND** TEXT_CATEGORY = "WRITTEN" **THEN** FORM = NONFINITE	870	643	0.338	0.469	0.575	0.550
IF NP_FUNCTION = DIRECT_OBJECT **THEN** FORM = RELATIVE	430	255	0.153	0.164	0.627	0.546
IF NP_FUNCTION = SUBJECT_COMP **THEN** FORM = RELATIVE	238	120	0.080	0.091	0.663	0.538
IF NP_FUNCTION = DETATCHED **THEN** FORM = RELATIVE	165	78	0.054	0.063	0.676	0.528
IF VP_TRANSITIVITY = CXTR **AND** NPHD_NUMBER = PLURAL **THEN** FORM = RELATIVE	83	31	0.025	0.032	0.724	0.521
IF VP_TRANSITIVITY = MONTR **THEN** FORM = NONFINITE	1372	1359	0.611	0.739	0.502	0.515
IF VP_TRANSITIVITY = CXTR **THEN** FORM = RELATIVE	271	168	0.098	0.104	0.616	0.513
IF SPEAKER_AGE = "66+" **THEN** FORM = RELATIVE	766	655	0.318	0.293	0.539	0.510
IF TEXT_CATEGORY = "WRITTEN" **THEN** FORM = RELATIVE	1125	1106	0.499	0.430	0.504	0.502

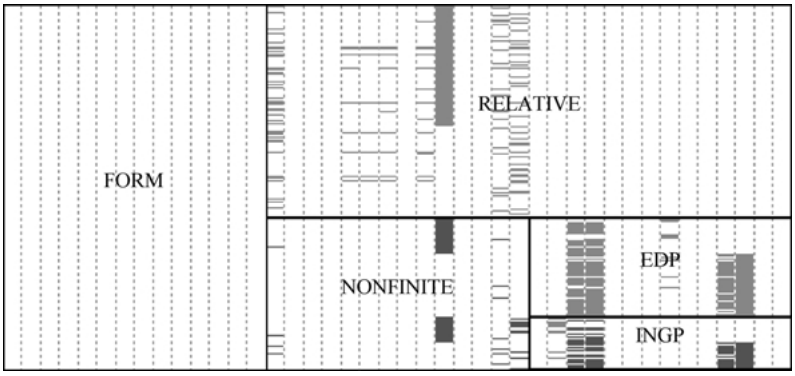


Figure 11. Coverage map for postmodifying clause form (where $u > 0.6$).

Acknowledgments

This paper would not have been possible without the efforts of the many linguists who constructed the ICE-GB corpus. Thanks are also due to Nigel Shadbolt, James Cupit, Tom Lavelle, Bas Aarts and the anonymous referees for their useful comments on this paper.

Notes

1. Parsers tend to asymptote in performance on unrestricted naturally occurring text at around 75% coverage and 70% accuracy (cf. Briscoe, 1996: the precise figures will vary according to the degree of sophistication and detail in the parse). It is very difficult to raise the performance of parsers beyond this point. Natural language is (a) highly ambiguous and (b) highly generative (new constructions keep appearing). Moreover, in order to disambiguate the syntax of a particular utterance, aditional semantic and pragmatic knowledge is often required.
2. More information about ICE and ICE-GB is available from http://www.ucl.ac.uk/english-usage/ice-gb/. The ICECUP software is freely available from this site, together with a sample of 20,000 words of parsed speech and writing from the ICE-GB corpus. The complete ICE-GB corpus used in this experiment is available on CD-ROM at cost from the Survey of English Usage.
3. Most of the KA support for the correction process (Wallis and Nelson, 1997) was developed late in the project lifecycle. Moreover, ICECUP was only available to support cross-sectional correction in the last year of an eight year effort (Wallis, 1999a). Late developing query systems can be highly revealing of inconsistencies and infelicities in the earlier analysis.
4. We refer to them as 'fuzzy' to encourage linguists to think about specifying inexact relationships and elements. They are *not* "fuzzy" in the sense of fuzzy logic (Zadeh, 1965). Online information about FTFs, including the methodology of simple experiments, is available at http://www.ucl.ac.uk/english-usage/ftfs.
5. The component nodes are as follows. A determiner phrase (DTP) followed by an adjective phrase (AJP) acting as an NP premodifier (NPPR), followed by a noun phrase head (NPHD) realised by a noun (N). Figure 3 indicates an example in the corpus: *our prime text*.
6. Additional terms: PU = parse unit, CL = clause, VP = verb phrase, SU = subject, VB = verbal, MVB = main verb, PRON = pronoun, DTCE = central determiner, ADJ = adjective, AJHD = adjective phrase head, A = adverbial, AVP = adverb phrase, ADV = adverb, AVHD = adverb phrase head.

7. For examples see http://www.ucl.ac.uk/english-usage/ftfs/experiment.htm.
8. UNIT permits any variable (source or target) in the dataset to be described in the form of a discrete, acyclic (set-subset) hierarchy of mutually exclusive terms. Examples of such variables are given in Table 2 and figure 9. In addition, UNIT can detect *refinement rules* (see note 14) for a hierarchical target variable.
9. Like many similar algorithms, UNIT reports aggregate statistics but not the actual set of cases covered by each rule (UNIT does have a graphical 'coverage map' facility—see Appendices—but this does not explicitly identify cases). Abstraction generated a new dataset from the corpus but cases in this dataset were not related to the corpus (recall also that a random 50% subsampling was used). Finally, ICECUP does not perform case resolution in the same way as our abstractive process.
10. In this study, we exclude the third nonfinite type, *infinitive*, because the correspondence between the relative clause and the nonfinite/reduced relative is less clear-cut. Unlike *-ing* and *-ed* clauses, infinitives permit correspondence with relative clauses in which the pronoun is not always a subject. It may, for instance, be an object: the man to *see* is Mr. Brown ∼the man who(m) you should see... (c.f. Quirk et al., 1985, pp. 1265–6).
11. Similar proposals have been made by, for example, Wu (1995), with respect to Quinlan's (1993) C4.5. ML is usually characterised as employing heuristic rather than statistically-pruned search.
12. C4.5 can obtain independent rules from data using a postprocess called C4.5RULES. The results are non-exhaustive due to C4.5's divide and cover strategy (Quinlan, 1993). Whilst UNIT will offer multiple possible explanations for a single phenomenon, C4.5RULES will tend to obtain a single 'best' explanation.
13. Collated from tables of mean frequency counts in Biber, 1986 (Appendix III, p 247), where *-ing* and *-ed* postmodifying clauses are referred to as "WHIZ deletion" relatives.
14. A refinement rule is a rule that includes a more general value of the target concept in the precondition, e.g., 'IF FORM = NONFINITE AND VP_TRANSITIVITY = INTR THEN FORM = INGP'. UNIT can exploit hierarchical structuring in the target variable to generate refinement rules.

References

Aarts, B., Nelson, G., and Wallis, S.A. 1998. Using fuzzy tree fragments to explore English grammar. English Today, 14:52–56.

Abeille (ed.) 1999. Journées ATALA sur les corpus annotés pour la syntaxe—Treebanks workshop, Paris: ATALA.

Biber, D. 1988. Variation across speech and writing. Cambridge: Cambridge University Press.

Brill, E. 1992. A simple rule-based template tagger. In Proc. 3rd International Conference on Applied Natural Language Processing, Trento, Italy. Association for Computational Linguistics, New Jersey, pp. 152–155.

Briscoe, T. 1996. Robust Parsing. In Survey of The State of the Art in Human Language Technology (on-line document), R.A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zoe (Eds.). http://cslu.cse.ogi.edu/HLTsurvey, (+/ch3node9.html).

Burnage, G. and Dunlop, D. 1992. Encoding the British National Corpus. In English Language Corpora: Design, Analysis and Exploitation, J. Aarts, P. de Haan, and N. Oostdijk (Eds.). Amsterdam: Rodopi.

Clark, P. and Niblett, T. 1989. The CN2 Induction Algorithm. Machine Learning, 3:263–283.

Corbridge, C., Rugg, G., Major, N.P., Shadbolt N.R., and Burton, A.M. 1994. Laddering: Technique and tool use in knowledge acquisition. Knowledge Acquisition, 6:315–341.

Cupit, J. and Shadbolt, N.R. 1996. Knowledge discovery in databases: Exploiting knowledge level redescription. In Advances in Knowledge Acquisition. Proc. 9th European Knowledge Acquisition Workshop, EKAW '96, N.R. Shadbolt, K. O'Hara, and G. Schreiber (Eds.). Berlin: Springer Verlag. pp. 245–261.

Fang, A.C. 1996. Automatically Generalising a Wide-Coverage Formal Grammar. In Synchronic Corpus Linguistics, C. Percy, C. Meyer, and I. Lancashire (Eds.). Amsterdam and Atlanta: Rodopi, pp. 131–146.

Fayyad, U. 1997. Editorial. Data Mining and Knowledge Discovery, 1:5–10.

Greenbaum, S. (Ed.) 1996. Comparing English Worldwide: The International Corpus of English. Oxford: Clarendon Press.

Haan, P. de. 1986. Exploring the Linguistic Database: Noun Phrase Complexity and Language Variation. In Corpus Linguistics and Beyond, W. Meijs (Ed.). Amsterdam: Rodopi, pp. 151–165.

Haberman, S.J. 1978. Analysis of Qualitative Data: Introductory Topics, Vol. 1, New York: Academic Press.
Huddleston, R. 1984. Introduction to the Grammar of English. Cambridge: Cambridge University Press.
Kondopoulou, P. 1997. Famine in Aigio: Analysing Memories of the Greek Famine. MA Dissertation, University of East London.
Lakatos, I. 1981. History of Science and Its Rational Reconstructions. In Scientific Revolutions, Oxford Readings in Philosophy, I. Hacking (Ed.) (1981). Oxford: OUP, pp. 107–127.
McEnery, A. and Wilson, A. 1996. Corpus Linguistics, Edinburgh: EUP.
Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, M., Ferguson, M., Katz, K., and Schasberger, B. 1994. The Penn Treebank: Annotating predicate argument structure. In Proc. Human Language Technology Workshop. San Francisco: Morgan Kaufmann.
Mooney, R.J. 1997. Inductive logic programming for natural language processing. In Proc. 6th International Workshop on Inductive Logic Programming, Berlin: Springer Verlag, pp. 3–21.
Mooney, R.J. and Califf, M.E. 1995. Induction of first order decision lists: Results on learning the past tense of English verbs. Journal of Artificial Intelligence Research, 3:1–24.
Muggleton, S. 1998. Inductive logic programming: Issues, results and the LLL challenge (abstract). In Proc. 13th European Conference on Artificial Intelligence, ECAI-98. Chichester: John Wiley. p. 697.
Michie, D. 1986. On Machine Intelligence (2nd ed.). Chichester: Ellis Horwood.
Nelson, G., Wallis, S.A., and Aarts, B. in press. Exploring Natural Language: The British Component of the International Corpus of English. Amsterdam: John Benjamins.
Niblett, T. and Bratko, I. 1987. Learning decision rules in noisy domains. In Bramer, M.A. (Ed.). Research and Development in Expert Systems, 3:25–34.
Quinlan, J. R. 1993. C4.5: Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann.
Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. 1985. A Comprehensive Grammar of the English Language. London: Longman.
Shaw, M. and Gaines, B. 1987. An interactive knowledge elicitation technique using personal construct technology. In Knowledge Acquisition for Expert Systems: A Practical Handbook. A. Kidd (Ed.). New York: Plenum Press.
Wallis, S.A. 1993. Machine learning with knowledge. In Proc. MLnet Workshop on Scientific Discovery 1993, MLnet, pp. 123–131.
Wallis, S.A. and Nelson, G. 1997. Syntactic parsing as a knowledge acquisition problem. In E. Plaza and R. Benjamins (Eds.). Knowledge Acquisition, Modeling and Management. Proc. 10th European Knowledge Acquisition Workshop, EKAW '97, Berlin: Springer Verlag. pp. 285–300.
Wallis, S.A., Aarts, B., and Nelson, G. 1999. Parsing in reverse—Exploring ICE-GB with Fuzzy Tree Fragments and ICECUP. In Corpora Galore, Papers from 19th Int. Conf. on English Language Research on Computerised Corpora, ICAME-98, J.M. Kirk (Ed.). Amsterdam: Rodopi, pp. 335–344.
Wallis, S.A. 1999a. Completing parsed corpora: From correction to evolution. In Abeille, 1999. pp. 7–12.
Wallis, S.A. 1999b. Machine Learning for Knowledge Discovery, PhD Thesis. University of Nottingham.
Wallis, S.A. and Nelson, G. 2000. Exploiting fuzzy tree fragments in the investigation of parsed corpora, Literary and Linguistic Computing, 15:339–361.
Wu, X. 1995. Knowledge Acquisition from Databases. Norwood, NJ: Ablex.
Zadeh, L. 1965. Fuzzy sets. Information and Control, 8:338–353.