 Parsing in reverse – Exploring ICE-GB with Fuzzy Tree Fragments and ICECUP

Sean Wallis, Bas Aarts and Gerald Nelson

Survey of English Usage, Department of English, University College London, UK.

Abstract

This paper describes a new approach to searching for grammatical constructions in parsed corpora, such as the new British Component of the International Corpus of English (ICE-GB). The idea is to express queries as approximate diagrams of the grammatical constructions in question. These ‘Fuzzy Tree Fragments’ (FTFs) are then efficiently matched against the entire corpus. The software, ICECUP, identifies how each case has been identified and allows users to browse the results. FTFs can be rapidly constructed from the grammatical analysis in the corpus, using a ‘Wizard’. This cyclic ‘exploratory’ approach allows newcomers to a corpus to quickly learn how relevant aspects of the analysis scheme employed in the particular corpus have been applied, and to construct well-understood queries.

1. Introduction

Text corpora are problematic objects of study. We spend years constructing them, assiduously collecting data, transcribing and annotating text. Even with the aid of automatic algorithms, the cost of expertise and effort is high. Such effort necessarily increases with the number and sophistication of the levels of annotation employed, the size of the corpus, and the degree of human checking and correction required. The 1 million word British Component of the International Corpus of English (ICE-GB) has taken in the region of 25 person-years of effort. (See note 1.)

Given the inevitable construction cost, we should ensure that we have efficient and perceptive techniques for analysing our corpora. Such techniques are rather less developed than tools for corpus construction, for two reasons. In the first place, construction necessarily precedes analysis, and in the second, many automatic algorithms, such as taggers and parsers, have a general application outside corpus linguistics.

Central to any human intervention in a large body of data, whether a regular database or a corpus, is the notion of a query. A query is a way of capturing precisely what it is that one wishes to find in the data. A query system applies this query to the database and extracts the cases that match the query. Thus, a lexical query for a single word should find every case of that word in the corpus. One query might retrieve portions of text according to the sociolinguistic information in the corpus, another might match grammatical elements.

Corpus query systems are central to analysis, and should also be central to correcting the annotation. We believe that such systems should be organised primarily around the principle of exploration, rather than ‘batch filtering’. This means that queries should be intuitive and easy to specify, linguistically meaningful, flexible, and fast.

2. ICE-GB and ICECUP

ICE-GB is a one-million word corpus of adult British English from the period 1990-3 inclusive. It consists of 500 samples, each approximately 2,000 words in length (300 spoken samples and 200 written). A total of 32 text categories have been sampled, including face-to-face conversations, classroom lessons, broadcast news, student essays, academic writing, and fiction (Greenbaum, 1996).

The corpus was tagged and parsed using software developed by the TOSCA Research Group at the University of Nijmegen. The parsed corpus contains over 83,000 syntactic trees, in which each node is labelled for function and category. In addition, many nodes contain additional information. VP nodes, for instance, carry feature labels which indicate their transitivity and their form. The parsed corpus was then checked by a team of researchers working at the Survey of English Usage.

The ICE Corpus Utility Program, or ICECUP, has been designed for ICE and similarly parsed corpora. The previous version (ICECUP 2.0) was designed for the tagged version of ICE-GB and was described in Porter & Quinn (1996). Version 3 is a new and radically improved program designed for the new fully-parsed corpus. Central to ICECUP 3 is a query system based on expressing tree-like grammatical queries, called Fuzzy Tree Fragments or ‘FTFs’ for short. The main initial use of FTFs was in fact to help us correct the corpus.

For some fairly obvious reasons, FTFs share a diagrammatic representation with corpus trees and other grammatical query systems, such as Nijmegen University’s Linguistic Data Base (van Halteren & van den Heuvel, 1990) and tgrep (see note 2). Arguably, FTFs are halfway in complexity between LDB’s patterns and tgrep’s wild card structures. However, where ICECUP differs from these systems is principally in approach. ICECUP is a system designed from the outset to provide “hands on” access to the ICE-GB corpus, and FTFs have been specified in this context. The problem can be summarised by the following conundrum:

Exploring a parsed corpus forces a user to learn the annotation scheme employed, from the meaning of labels through to integrated parsing decisions. But if you don’t know what you are looking for (in terms of this annotation scheme) how can you specify a query to explore the corpus in the first place?

This apparent contradiction is solved practically by providing a platform for cyclic exploration, illustrated by Figure 1. Cyclic exploration means both refining queries by repeated experimentation, and using the corpus itself to construct queries. ICECUP offers a device called the “Create FTF Wizard”, which extracts part of a corpus tree structure and creates a grammatical query from it.
1. Corpus map

2. Browse text (S1A-002)

3. Browse tree (S1A-002 #1-2)

4. Create FTF Wizard

5. New FTF

6. FTF search (in progress)

7. Browse text (FTF query results)

8. Browse tree (matched with FTF)

Figure 1: Cyclic corpus exploration with ICECUP

This can then operate as a starting point for further exploration. In addition, ICECUP provides a variety of simple queries and a top-level ‘corpus map’ which summarises the organisation of the corpus.

3. Introducing FTFs

The simplest kind of Fuzzy Tree Fragment is shown in Figure 2. It is simply an empty node with some unspecified links. Nodes are depicted as consisting of three segments: the two upper quarters contain functional and categorical codes, the lower half contains a series of applicable features.

FFuzzy Tree Fragments are “fuzzy” in a number of ways: nodes can be incompletely specified, so that, for example, the functional role of an element or some of its features may be omitted. Secondly, the relationship between one node and another may be partially specified. Finally, FTFs are fragments, or branches of trees.

3.1. Two practical examples

Let us look at some practical examples. Figure 3(a) below specifies a query for conjoined elements that contain a notional direct object (‘NOOD’) immediately
below the node, while 3(b) introduces the requirement that there be an object complement, marked ‘CO’, which immediately precedes the notional direct object.

White lines in these figures indicate unspecified relations. In Figure 3(a) there are three specified requirements for a tree in the corpus to match the query.

1. There must be a node with the function ‘conjoin’ (CJ).
2. There must be a node with the function of ‘notional direct object’ (NOOD).
3. These two nodes must be adjacent in a parent-child relationship: the former should immediately dominate the latter.

One of the key ideas of FTFs is that it is easier to draw a diagram than state (e.g., in logic) a set of facts like this. As FTFs become larger, these benefits become more obvious. Thus, Figure 3(b) would require three further facts.

4. There must be a node with the function ‘object complement’.
5. This node must also be an immediate child of the ‘conjoin’ node.
6. This node must immediately precede, in the set of children, the ‘notional direct object’ node.

FTFs benefit from a diagrammatic representation in another way. The FTF is presented as a coherent whole, rather than as a set of disconnected statements. One aspect of this is parsimony. We can easily see that fact 5 is redundant, because it flows directly from facts 3 and 6. The main gain is intuition: FTFs are easy to learn by analogy with grammatical trees, and can capture the essentials of grammatical structure.

If we employ an FTF to search the corpus, we can quickly retrieve the set of matching cases, such as e.g., the tree in Figure 4, which matches the FTF in Figure 3(b), from a newspaper in the written part of ICE-GB (hence the line-breaks within words).

The FTF is employed, not merely to search the corpus, but also to highlight relevant parts of the text unit. First, the nodes matching the FTF are indicated. Second, the text under the FTF focus is also highlighted (see Figure 2). In Figure 3(b) the FTF focus was applied to the ‘conjoin’ node, so the entire text under this node is underlined in Figure 4. This focus does not restrict the search: rather, it specifies our preferred point of interest within the FTF.

The focus, or point of interest, is used for concordancing with FTFs. Figure 5 shows a portion of the complete set of matching cases (15 in 13 text units) for this FTF from ICE-GB.

3.2. Dimensions of fuzziness

The FTFs in the previous section were specified as having nodes immediately, or ‘intimately’ connected. However, often this is too restrictive. FTFs allow you to make the links between nodes more ‘fuzzy’. This can be done in two distinct ‘dimensions’.

a) Parent-child. Two nodes A and B which are connected as parent to child in an FTF are allowed to match two nodes in a corpus tree, provided A is the ‘ancestor’ of B (Figure 6(a)).

![Figure 4: A matching case for the FTF in Figure 3(b)](image1)

![Figure 5: A concordance display for the FTF in Figure 3(b)](image2)

![Figure 6: Two different dimensions of ‘fuzziness’ in FTFs (the white lines are emphasised for clarity)](image3)
b) Child-child. Two adjacent child nodes A and B in an FTF may match two nodes in a corpus tree as long as A is in the sequence precedes B (Figure 6(b)).

There are actually several different ‘fuzzy’ relationships that you can apply between two FTF children. If we want to, we can specify that the ordering between two sibling nodes is unimportant, which gives us four links.

- ‘immediately after’ (represented as a black, uni-directional arrow: ‘→’).
- ‘anywhere after’ (represented as a white uni-directional arrow).
- ‘just before or just after’ (represented as a black bi-directional arrow: ‘←’).
- ‘anywhere before or after’ (represented as a white, bi-directional arrow).

Child-parent relationships must be ordered, so the only option is immediately connected (‘parent’) or eventually connected (‘ancestor’).

These four child-to-child relationships imply something else. FTF searches are strictly grammatical. However, this is not always desirable, for lexical queries in particular. If one wishes to search for a simple two word sequence, for example, one would probably not wish to specify grammatical structure between the words. We therefore have two further inter-sibling link values: $<unknown>$, and one called ‘different branches’. While $<unknown>$ removes any restriction between the siblings, ‘different branches’ enforces the less weak restriction that the two children must appear on different branches of the tree (one cannot be the ancestor of the other).

3.3. Edges of fragments

As well as specifying links between two nodes, you can also make other simple topological statements about the position of a node, where appropriate. For example, you can state that your topmost node is at the root of the corpus tree, or you may specify that a node is the first child in a group of siblings, or that a child in your FTF with nothing specified under it, is, in fact, a ‘leaf’ node (this means

![An example FTF with edges specified](image1)

![A corpus case matching (a)](image2)

Figure 7: Specifying the edges in an FTF

that it meets the text at the bottom of the corpus tree). You may also specify the reverse, namely that a node is not a root/first child/last child/leaf.

‘Edge’ markers (see Figure 2) thus have three potential settings: ‘Yes’, ‘No’ and $<unknown>$ (the default). In order to present these edges coherently, edges marked as $<unknown>$ are drawn as white lines (see Figures 2-3), representing links to possible other nodes. Edges marked as ‘Yes’ mean that there is no possibility of any further elements. These are depicted by the absence of a line. Similarly, edges marked as ‘No’ mean that further elements are possible, and they are drawn as solid black lines extending from the FTF (Figure 7(a)). Edges further limit the matching of cases, and should be applied with care, lest they make the FTF too restrictive. However, sometimes their use is unavoidable, particularly in relation to text.

3.4. Text and tree fragments

Text elements can be freely introduced into FTFs. The right hand margin contains text unit elements. In the corpus, each leaf node in the tree annotates a single word (or, more correctly, ‘text unit element’ – this includes lexical and non-lexical items, such as pauses). In an FTF, we can add a word under any child node. The relationship between the node and the word is actually implied by the ‘leaf’ status of the node. If a node is a leaf, then it must connect directly to the word; if it is not a leaf, or it is unknown, then it is eventually connected.

Words are connected together in a chain with a single ‘next word’ link. This is similar to the ‘next child’ link, and offers almost the same set of arrows, except for ‘different branches’ which is not applicable to words. For completeness, there are also ‘first word’ and ‘last word’ edges, although these are seldom used.

A text-oriented search must be independent from any grammatical annotation on the text. However, as we have seen, FTFs are essentially a grammatical representation. How do we make the FTF ‘disappear’ so as to specify a clean text sequence like “in particular” in Figure 8(a)?

The principled answer is indicated by Figure 8(b). The two lower nodes are made to annotate the two words directly: they are given the ‘leaf’ status of ‘Yes’. This means that the upper node is tied to the root node. We then simply relate the nodes together by stating the minimum, namely, (a) that the leaf nodes have the root as their ancestor – which is always true – and (b) that the two leaf nodes have an $<unknown>$ relationship between them.

The only active relationship is the ‘inter-word’ one on the right hand side, which states that the word “particular” must immediately follow “in”. This is indicated by the uni-directional black arrow.

Naturally, using a grammatical representation for a non-grammatical query means that specifying the FTF in Figure 8(b) by hand could be problematic. However, the query interface in Figure 8(a) generates 8(b) automatically.
4. Conclusions

FTFs are simple, intuitive representations for specifying grammatical queries. Their simplicity is essential because they allow the user to concentrate on learning the grammatical scheme employed in the corpus, rather than having to learn to ‘program’ the query system. Just because specifying a search may be simple, this does not mean that the query cannot be effective or ultimately used in a complex way. Thus in ICECUP, FTFs may be combined with other kinds of query, such as sociolinguistic queries, using propositional logic.

The claimed ‘intuitiveness’ of FTFs rests partially on their simplicity, and on the fact that they look like corpus trees, with some information omitted.

The real test is in use. ICECUP and ICE-GB are available to the linguistic community from the Survey of English Usage (http://www.ucl.ac.uk/english-usage/). A ‘Sample corpus’, consisting of ten fully-parsed texts (20,000 words) from ICE-GB, is available for free download from this web site. This includes the full ICECUP software, including FTFs and complete help information.

Naturally, we recognise that FTFs are extendable in a number of ways, and may be applied to a number of other tasks aside from linguistic research. We encourage readers to give us feedback on the existing system, and we will be reporting on further developments at future ICAME conferences.