Corpus-based test suite generation

Lorna Balkan, Frederik Fouvry.

Report to LRE 62-089 (D-WP5.2)

Test Suites for Natural Language Processing (TSNLP)

University of Essex

October 1995

Summary

Title	Corpus-based test suite generation
Project	LRE-62-089
Document ID	D-WP5.2
WP	Test Suite Design
Date	October 1995
Authors	Lorna Balkan, Frederik Fouvry
Reviewers	The consortium
Status	Final document

Contents

Corpus-based test suite generation

1 Introduction

The notion of test suite we have been assuming in the project up until now is that test suites are artificially constructed data. We noted how this allows one maximum control over data construction. We contrasted this with testing via test corpora where the occurrence of phenomena subtypes and combinations is always arbitrary. However, we noted that one criticism that can be levelled against the test suite approach is that the data can be contrived and never actually occur in real life. In this work package we explore a compromise solution—the derivation of test suites either automatically or semi-automatically from corpora. We took as our basis an input paper on corpus studies by Dauphin and Lux ([6]) and a small trial we conducted on syntactically analysed corpora (see section 3.3).

2 Corpus Analysis and Test suites: existing work

Corpora are a useful resource for evaluation. They can be used in a number of ways. First, they can be used to assign weightings to items in test suites: the higher the frequency of a test item in a corpus, the higher its weighting should be. Secondly, (representative) extracts of corpora, "test corpora", can be used to complement test suites. In Balkan et al. ([2] we discussed some of the ways in which information in test corpora can be used to complement information in a test suite. They provide information on lexical, semantic and pragmatic phenomena, which are phenomena not well suited to the test suite method. They also provide examples of extra-linguistic phenomena (acronyms, etc.). Thirdly, and this is the object of inquiry here, they can provide a basis for the construction of test suites themselves. We look below at three corpus-based approaches to test suite construction. Approaches range from lifting test items directly from test corpora (Lehrberger and Bourbeau) to using corpora (as opposed to grammar books, say) as inspiration for the writing of test items. Section 2.3 describes work that has been carried out by consortium members, within the framework of other projects.

2.1 Lehrberger and Bourbeau

The work of Lehrberger and Bourbeau ([13]) has already been described in [2]. They describe a methodology for creating evaluation material for machine translation systems from the point of view of the user. Test suites are derived directly from corpora, i.e. not artificially constructed. They stress that the sentences should contain a representative set of sentences from a given domain, without, however, giving concrete suggestions about how to choose this set of sentences beyond studying the linguistic characteristics of the texts from which they are to be taken. Once sentences have been put through the system, translation errors are noted and classified according to linguistic phenomenon, where a distinction is made between phenomena specific to the comprehension of the source language and those specific to the transfer of this comprehension into the target language.

The distinction between test suites and test corpora is not explored in any length, but presumably the latter imposes the constraint that the sentences be a naturally occurring sequence of sentences.

2.2 Lepage

Lepage ([14]), like Lehrberger and Bourbeau, describes a methodology for writing test suites for machine translation systems. The interest was mainly in the translation of noun phrases, and the work described concentrates on the translation of the French preposition *de* in noun phrases.

The starting point for the study was the French corpus, from which all occurrences of the form *de* i.e. *du* = *de* + *le*, and *des* = *de* + *les* were automatically extracted. The English translations of the word *de* were manually marked in the English raw translation produced by the machine translation system. Not all of the constructions containing *de* were noun phrases, of course. On the basis of this analysis two test suites were written. One, the "selection" test suite, contained all instances of structures containing *de* that were not noun phrases. This test suite was to probe the system's ability to recognize such structures as distinct from noun phrases. Two versions of this test suite were produced—a "simple" version derived by simplifying the corpus sentences illustrating the phenomenon, and a "complex" version, where more translation difficulties (such as coordination) were introduced into the simple test items. In addition to the "selection" test suite, a second type of test suite was produced, which concentrated on noun phrases. They were categorized according to their argument structure: e.g. *comportements des matériels* where the second noun (*matériels*) corresponds to the arg1 (subject) of the predication, is contrasted with *simulation des comportements* where the second N corresponds to the arg2 (object) of the predication. Nouns were also classified according to their semantic class. Data is then gathered on the usual translations of the noun phrases, based on their argument structure and their semantic class information.

Lepage's approach uses corpora as a means of obtaining initial data, but the test suites produced are the result of the test suite writer's linguistic knowledge to simplify and classify the data, using syntactic, semantic criteria, as well as knowledge about translation difficulties.

2.3 Aerospatiale

The Aerospatiale approach to test suite construction has already been described in Balkan et al. ([2]). Here we summarize their methodology and cite complementary work on corpus analysis carried out by Dauphin ([5]).

Regnier and Dauphin ([16]) describe a methodology for test suite construction for machine translation from an industrial user's point of view. Text is submitted to an MT system and the output compared with proofed text. Errors are then classified according to type:

- lexical/terminological
- morphological
- syntactic
- segmentation

The corpus sentences containing these errors constitute the initial test suite. From these, however, "generic" test suites are produced, where:

- vocabulary is simplified and reduced
- phenomena are isolated

Individual phenomena in the input sentences are isolated and for each phenomenon a sentence as short and simple as possible is built to exemplify it, maintaining where possible, the original vocabulary (particularly if the phenomenon is likely to be vocabulary dependent) or the original layout. In a further stage, for each phenomenon, a set of test sentences is written that are intended to illustrate all possible linguistic variations of the phenomenon (in terms of lexical, morphological, syntactic, semantic and pragmatic variations). As with the Lepage test suites, Regnier and Dauphin use the corpus initially to provide empirically encountered problems, but the test suites that are derived are based on linguistic analysis and classification.

Dauphin ([5]) reports on corpus studies she carried out in order to identify the linguistic phenomena in a corpus with a view to writing a series of test suites to test these phenomena. The aim of the study was to establish the type of linguistic problems an MT system would need to treat in a particular corpus in order to translate it. The work was partially automated using the corpus which had been lemmatized and tagged with a part of speech tagger.[1] The automatised part of the study was carried out within the framework of the EUREKA GRAAL project.

The study was intended to identify the grammatical, lexical and terminological phenomena in the corpus, as well as extra-grammatical phenomena such

[1] The tool is called AlethCorp and was developed by GSI-Erli, but is not publicly available.

as lists, parentheses, chapter headings, etc. The identification of the latter type of phenomena was facilitated through the use of SGML markup. The lexical study paid particular attention to the identification of abbreviations, initials, and acronyms, as well as of proper names and foreign words, all of which are known to be problematic for MT systems. Lemmatisation of the corpus and the part of speech tags facilitated the study of lexical phenomena. Terminology searches were done on the basis of known patterns of terms (e.g. `N + N`, `N` *à*, etc.).

Grammatical structures were harder to identify with a part of speech tagger and were mainly done manually. Phenomena of interest included conjunction, relative clauses, infinitives, indirect questions, tense, diathesis, and verb complementation. Of particular interest were `NP`s and adverb phrases. Anaphoric pronouns were also studied to see whether there were any regularities in the position of the antecedent (a prior study found that 90 percent of the antecedents were the direct object of the preceding sentence, a fact that can be usefully employed in system heuristics).

Corpus analysis not only allows the identification of linguistic phenomena present in a corpus, but the degree of complexity of constructions present in the corpus. Studies at Aerospatiale showed that most translation errors arose from constructions containing a range of phenomena, and thus test suites were written in the first instance to test individual phenomena, which could then be progressively combined to test rule interaction.

To summarize, corpus analysis (whether automatic or not) is considered useful for system developer and system evaluator alike in that it allows the identification of the exact characteristics of the corpus. This allows the system developer to tune the system's grammatical and lexical coverage to the user's needs, and it allows the evaluator to concentrate on the evaluation of phenomena that were known to be present in the corpus. It even suggests the type of heuristics (for anaphora resolution, for example) that the system developer could usefully incorporate into his system. However, corpus sentences by themselves need further processing, and decomposition into the individual phenomena that they contain.

Dauphin notes that for a multilingual application such as machine translation, what would be useful, in addition to analysis on monolingual texts, is analysis on aligned, bilingual texts.

3 Automatic test suite extraction from corpora: some experiments

We begin by summarizing some of the different types of corpora that can be exploited for test suite generation. (We restrict ourselves to the discussion of monolingual corpora, although, as we noted above, for multilingual applications, the availability of bilingual corpora and tools (such as alignment tools) would be a useful resource.)

- unanalyzed corpora
- corpora tagged for part of speech (PoS) or lemmas:
  Examples of PoS taggers are the CLAWS tagger, ([9]) which was described in Estival *et al.* ([8]).
- syntactically analysed corpora:
  Examples include the hand-parsed Penn Treebank ([15]), the Linguistic Database (LDB) and the SUSANNE Corpus ([18]).

We discuss the usefulness of these various types of corpora for test suite extraction.

3.1 Unanalyzed corpora

Concordance programs can be used even on raw corpora. Concordance programs typically search for a word and retrieve the word together with the context in which it was found. Concordance programs have long been used in the field of lexicography. As far as test suite construction is concerned, however, they are of limited use, containing as they do no syntactic information.

3.2 Tagged corpora

A lemmatized corpus, where each word has been assigned its base form or lemma, facilitates searches on particular words, if one wants to abstract away from the particular form of that word (e.g. if one wants to search for all forms of a verb, and not just the base form, for example).

A tagged corpus is of potentially more interest to the test suite builder. Pattern matching tools can be used on tagged corpora to find instances of a tag or a sequence of tags. Search tools such as these have several applications. They have been used, for example, in the ET-10/63 Project Statistical, text-corpora based complements for Eurotra: Terminology, Lexicon and Preference, and the ET-10/75 Collocations Project, The Lexicalisation of Semantic Operations for the extraction of terminology and collocations respectively. And, as we saw above, they were also used by Aerospatiale in its corpus analysis work. There are several problems with using this type of tool for test suite extraction. First of all formulating searches by means of regular expressions is not always user-friendly. Secondly, some constructions are inherently difficult to recognize using tag patterns alone. An example is reduced relatives (*The man he saw sold the house*). Unlike unreduced relatives, there is no *wh*-word or *that* to signal the presence of a relative clause. Other constructions such as purpose adjuncts and infinitival complements may be hard to distinguish without higher level information: *John requires Bill to complete the project* needs contextual information to know whether the infinitive is to be interpreted as a purpose adjunct where (*to* is equivalent to *in order to*) or as an infinitival complement. Thirdly, errors

may arise due to inaccurate tagging. For example, participles are sometimes hard to distinguish from adjectives and nouns:
John has [NP rented furniture].
John was [AP smashed (drunk)] last night.
John's favourite activity [VP is watching TV].
(examples from Brent ([3])).

Dauphin and Lux ([6]) discuss the particular problems associated with the identification of terminology, which was a focus of their corpus analyses. Well-known problems of terminology identification include:

- composition
  For example, *geostationary telecommunication satellite* consists of not one but two terms, namely *geostationary satellite* and *telecommunication satellite*.

- coordination
  For example, *landing and roll out control units* can be factorised into two terms *landing control unit* and *roll out control unit*.

- insertion
  For example, the term *electrical device* is contained within *electrical small device*.

Dauphin and Lux further discuss the problems that non-standard syntax poses for corpus analysis. Examples include the omission of articles (in particular before acronyms) and prepositions (*code évolution page* for *code d'évolution page*) or non verbal phrases (*Ensemble stocké en containers étanches et non pressurisés*).

3.3 Parsed corpora

Syntactically analysed corpora should have more potential for the test suite writer than corpora tagged for PoS. However, there are few of them. Of note is the *University of Pennsyvania Tree Bank*, which has been used in the evaluation of parsers, for example in the Parseval exercise ([10]). In Parseval a simplified annotated sentence of the UPenn Treebank provided a "standardized" parse with which to compare system output.

The Upenn Tree Bank project ([15]) was developed in two phases. The first phase produced a 4.5 million word PoS tagged corpus, the second (and current) phase is producing a syntactic annotation of a 3 million word corpus ([12]). The PoS annotations are a reduced set of the Brown Corpus PoS annotations. The reduction consists of removal of redundant information, assigning a more important role to the functions (that form a start for a bracketed corpus), and assigning multiple tagging in case of ambiguity. Overall there are 36 PoS-tags, and 12 punctuation tags. The annotation was done automatically, and this output was checked manually.

The second phase of the project has added 20 function tags (among others adverbials and logical/surface subject), and some additional null elements. These innovations, plus increased consistency in the annotations, are a response to the needs expressed by the users of the results of the first phase.

The automatic parse is non-deterministic, and assigns only one parse (if necessary partially). To keep the structures as general as possible, the parses are kept "skeletal".

Other syntactically analysed corpora and associated tools include:

- SUSANNE Corpus (using SUSIE)
- The Nijmegen Corpus (using Linguistic Database)

to which we had access in the project. From these we conducted some trials on extracting test suites. We describe each in the sections that follow, giving a description of the corpora and the tools to analyze them. We then report the results of our investigation. Our purpose was not to conduct a comparative study of the two corpora and tools. Rather we were interested to know what type of corpus annotations and tools would be necessary for test suite extraction. We concentrated amongst other things in evaluating the following characteristics:

- the extent and type of syntactic labels (e.g. syntactic function labels, syntactic category labels)
- the attribute information on syntactic labels
- any additional information in the corpus, e.g. indication of ungrammaticality and/or ambiguity
- the ability of tool to perform searches at different levels (e.g. at attribute level, syntactic category level, syntactic function level, or any combination of these)

3.3.1 The SUSANNE Corpus

The SUSANNE Corpus ([18]) is a subset of the Brown Corpus of written American English which has been annotated (mainly by hand) with a scheme based on the Lancaster-Leeds Treebank ([9]) but including specifications of logical as well as surface structure.[2]

It contains six data fields, visualized as columns:

Location	Status	Word tag	Word	Lemma	Parse
`A01:0010b`	-	`AT`	`The`	`the`	`[O[S[Nns:s.`
`A01:0010c`	-	`NP1s`	`Fulton`	`Fulton`	`[Nns.`
`A01:0010d`	-	`NNL1cb`	`County`	`county`	`.Nns]`
`A01:0010e`	-	`JJ`	`Grand`	`grand`	`.`

[2]See also [11, 12].

```
A01:0010f       -       NN1c    Jury    jury    .Nns:s]
A01:0010g       -       VVDv    said    say     [Vd.Vd]
(...)
```

(The annotations for the whole sentence of which this is a fragment is given in Appendix 3).

The sixth column, the parse field is "the central raison d'être of the SUSANNE Corpus" ([19], p.178). Labelled brackets are used to represent the parse tree. The leaf node tags are not directly indicated, instead a dot ("`.`") is used as a shorthand to indicate "`[`<word tag (col.3)>`]`". As there is a leaf node on each line, there is a dot in the sixth column of each line. Non-terminal nodes can contain up to three types of information, indicated in the bracket labels:

- Form tags
- Function tags
- Index

Form tags are always present (except for nodes immediately dominating traces). There are different levels of form tags:

- Word level form tags
  There are 352 distinct word tags, based on the Lancaster tag set (see Appendix B of [9]). (Word level form tags begin with two capital letters, while all other form tags begin with one capital and contain no further capitals).
- Phrase level form tags
- Clause level form tags
- Root level form tags

Function tags divide into complement and adjunct tags. They are assigned to all immediate constituents of clauses, except for their verb-group heads and certain other constituents for which function labelling is inappropriate. Indices are used to indicate referential identity.

A representative sample of phrasal, clause level, root level form tags and function tags is given in the Appendix 1 and are taken from [19], p.169–187.[3]

A How to interpret the annotations in SUSANNE

The information associated with a field on each line is appended to form a string. For example, the appended fields 6 (parse) and fields 3 (word tags)

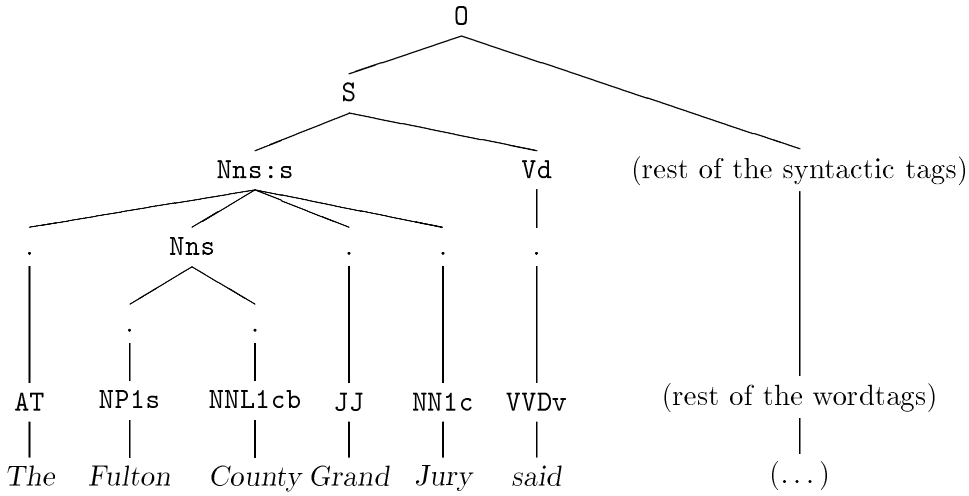
[3] A list of word tags was made available to us by the kind permission of Geoffrey Sampson and is shortly to be published in [18].

for the sentence fragment quoted above (*the Fulton County grand jury said*) are as shown below:

`[O[S[Nns:s. [Nns. . Nns]. . Nns:s][Vd. Vd]` (field 6)

`AT NP1s NNL1cb JJ NN1c VVDv` (field 3)

Note that category labels occur twice – at the beginning and end of square brackets.[4] As mentioned above, the dots represent leaf nodes: the word tags are placed under them. So the tree of the sentence fragment above looks like this:



The tags (listed in the Appendix 1) have subclasses indicated by lowercase letters. The lowercase letters (from any one category) may be combined and in this way the full range of the tag set can be created. Take, for example, `Nns:s` from the sentence fragment: `N` is a phrase level (form) tag, which has two subcategory symbols, viz. `n` for proper noun and `s` for singular. The label `Nns:s` contains additional information in the form of a function tag `s`: thus the node labelled `Nns` in this tree functions as the subject.

B The tool

We conducted investigations of the corpus using the SUSIE toolkit (version 2.0 alpha, [4]). SUSIE is a pilot program investigating the possibility of corpus-searching using a DCG-based parser.[5]

Searches can be done by matching sequences of words or sequences of tags (word tags or syntactic tags) using regular expressions or by the use of "grammar rules". Grammar matching is claimed to be the most powerful feature of SUSIE. To do grammar matching, a grammar file is written with rules are in the form of:

[4]The double occurrence of the labels brings along a certain redundancy, but renders searches more robust and easier to perform, for machine and human user alike.

[5]A much-enhanced program suite is currently under development and is intended for release into the public domain in the future. The working name for this project is CORSET (Corpus Search Toolkit).

```
lhs --> rhs
```

where `lhs` is any symbol, and `rhs` is a list of one or more symbols, e.g.

```
s --> np vp
```

The choice of symbols is completely abitrary, but "terminal" symbols should match the word-tags in the corpus. For example:

```
vppasttense --> VBD np
```

where `VBD` is a tag for a past tense verb in the corpus. "`*`" meaning "zero or more of the preceding symbol" and "`+`" meaning "one or more" can be incorporated into the rules. E.g.

```
n --> n+
```

And for `np`s a rule like

```
np --> det n pp*
```

is allowable.

Grammar matching is done on the basis of the `rhs` of the grammar rules.

SUSIE highlights the instance of the matching pattern on line, but this (the indication of the match) cannot be stored. The user is thus required to edit the output.

A query in SUSIE is formulated as follows:

```
syntax = "<regular expression>"     for syntactic tags (col.6)
words = "<regular expression>"      for words (col.4)
tags = "<regular expression>"       for wordtags (col.3)
grammar = "<lhs of a grammar rule>" for grammar rules (using wordtags)
```

Those can be preceded by `not` or by `#n`, where `n` is the number of a preceding query, followed by `and`, `or` or `xor`.

C Results

Information in corpus

- Given the possibility of forming subcategories from category labels, the range of syntactic tags is very extensive. We discuss some examples below.

  The following different types of coordination are distinguished:

  - subordinate conjunct introduced by conjunction (annotated by `+`),
  - subordinate conjunct not introduced by conjunction, `-`

- appositional element, `@`
- coordinate structure acting as first conjunct within a higher coordination `&`

Coordination is possible at the word level and at higher levels.

- Where nouns are unambiguously marked for plural in the language, they have a plural annotation (`2` for word tags, `p` for syntactic tags). However, where their morphological form is neutral with respect to number (e.g. *fish*), then they have no such annotations. This applies to verbs as well:
  Dr. Barnes said that there seemed to be feeling that evacuation plans, even for a high school where there were lots of cars "might not be realistic and would not work".
  `[O[S[Nns:s..Nns:s][Vd.Vd][Fn:o..[Vd.Vd][Ti:s[s152.s152][Vib.. Vib]Ti:s][Ns:S152.[Fn.[`*Np*`:s....[P.[Ns:151...[Fr[Rq:p151.Rq:p 151].[Vwb.Vwb][`*Np*`:s.[Po..Po]`*Np*`:s]Fr]Ns:151]P]`*Np*`:s].[`*Vdceb*`.. .`*Vdceb*`][J:e.J:e][Fn+.[Vdce...Vdce]Fn+].Fn]Ns:S152]Fn:o]S].O]`

  Thus *plans* and *cars* are marked for plural (`Np`, indicated in italics), but not *might* (`Vdceb`, in italics), which in this case is also plural. This means that one is unable, for example, to search for all plural verbs, preceded by plural noun phrase, as one might want to do, for instance, if one were collecting data for subject-verb agreement.

  Verb phrases are however marked for tense (`d` indicates a verb group beginning with the past tense), modality (`c` indicates that the verb group begins with a modal verb) (cf. Appendix 1, phrase subcategories)). The same information is also present in the word tags (e.g. *might* (`VMd` where *M* means modal, and *d* means past form). Lacking, however, is any information about subcategorisation, beyond the symbols `t` (transitive) and `i` (intransitive) on word tags. The user has to derive the information from the presence of function tags (`:s` (logical subject), `:i` (indirect object) , . . .) in the daughters of the verb phrase.

- Passive verb groups are indicated by `Vp` (e.g. *The couple was* married *Aug. 2, 1913.*
  `[O[S[Ns:S..Ns:S][Vs`*p*`..Vs`*p*`][N:t....N:t]S].O]`)
  and verb particles (e.g. in *His political career goes* back *to his election to city council in 1923.*
  `[O[S[Ns:s...Ns:s][Vz.Vz][R:`*n*`.R:`*n*`][P:q.[Ns..[P.[Ns..Ns]P][P.. P]Ns]P:q]S].O]`) have the function tag `n` and are therefore easy to retrieve.

- There is a lot of semantic information present in the tags (see e.g. `ND` (direction noun, e.g. *North*, `NP` (proper name), `NPD` (day of the week) etc.), especially in the word tags. Titles are indicated both in the word tags and in the syntactic tags (e.g. by `YBL` and `Oh`) respectively for headings), so they are easy to retrieve. This is very useful for lexical replacement.

- An interesting feature of SUSANNE is the indication of ungrammaticality. Errors in the text are indicated in the status field.

Search techniques

- It is quite easy to look for attributes where they are systematically present in the tagset, e.g.
  All the subject (`:s`) plural (`p`) nouns (`N`), immediately (`\.`) followed by a relative clause (`Fr`):

  ```
  syntax = "N[^:[.]*p[^:[.]*:s[^.]*\.\[Fr"
  ```

- Other cases are not so straightforward, however. There is no specific tag for reduced relative clauses. It requires the formulation of the following regular expression:

  ```
  syntax = "Fr\[N[^qv.[:]*"
  ```

  (i.e. a relative clause (`Fr`), of which the label of the first daughter is an `N` and contains no `q` or `v`).

- It is possible to use intersections and unions of search sets, but only on utterances as a whole. For example, if one looks for plural indirect objects using the command:

  ```
  syntax = ":i"
  #1 and syntax = "Np"
  ```

  one will retrieve the sentences with either a plural subject or a direct object, and not both, as required. This behaviour is based on the "match effects" in SUSIE: as soon as a match is found, the program starts checking a new sentence.
  The correct formulation of this search is as follows:

  ```
  syntax = "Np:i"
  ```

- The grammar rules, described earlier, use only word tags. We give an example grammar in Appendix 2. A certain amount of care has to be taken in writing grammar rules. Formulating a search for nouns preceded by the determiner *those* produced the result *Those union*. The error arose because the word *union* was actually part of the compound *union members*. Thus, as with all corpus analysis, results have to be inspected by hand. However, we feel that the usefulness of the grammar search facility could be enhanced if one were able to combine word tags and higher level tags in the grammar rules. The grammar as it is now gains in usefulness in combination with set operations on search results. The definitions in the grammar can thus be refined considerably. One must bear in mind however the drawbacks of these operations (cf. *supra*). Given the availability of syntactic tags, one is disinclined to spend a lot of time writing grammar rules.

3.3.2 The Nijmegen Corpus

This corpus was available to us in smaller sample corpora, provided by the authors of the Linguistic Database (henceforth LDB, [20]). LDB is a corpus analysis tool, and extracts of the Nijmegen Corpus were readily accessible with this tool.[6] We give a description of it in the next section. We were also provided with a list of the function and category names used in the corpus.

For a view of the plain corpus files, see Appendix 5. The sample corpora we tested offered generally much less information[7] than SUSANNE (except for subcategorisation, which makes a finer distinction between various kinds of transitive and intransitive verbs—see Appendix 4). The Nijmegen Corpus analyses however are generated with the help of a parser, from a (mostly) context free grammar, and offer therefore perhaps greater consistency than SUSANNE provides.[8].

The corpora we used had two tag sets: category and function.[9] The definition of the sets is circular, i.e. the categories are defined in a grammar in which a function can be rewritten as a category and a category as a sequence of functions. Functions exist at every node, unlike in the SUSANNE Corpus, where they are only used at sentence and clause levels.

A The tool: LDB

The tool allows for three kinds of tag: function, category and attribute tags. Function tags and category tags from the corpus are listed in Appendix 4. As the tool is a database interface, it does not presuppose any specific formats or tags for a corpus. The names do not form part of the LDB terminology ([20], p.11). This decision is up to the writer of the corpus to be used by LDB. The attribute tags are meant to contain a list of properties, being anything that does not fit into the category or function slot, which may only contain one value ([20], p.4). The slots used depend on the grammar used to analyse the corpus, e.g. which features will fill the attribute slot.

The tool allows for a more sophisticated search in a corpus. LDB is more

[6] The program is the result of the project "LDB, a linguistic database" at the university of Nijmegen (NL).([20], p.2)

[7] 51 functions (6 of which are punctuation and catch-all cases) and 95 categories (42 of which are verbal tags). We are informed, however, that the most recent Nijmegen analyses (to which we did not have access) are more detailed: 80 functions, 70 categories and more than 100 optional attributes for nodes, e.g. `V(trans,pastp)` for a past participle of a transitive verb.

[8] See also [11, 12]

[9] The terminology "category" and "function" is that of LDB. We leave it open here how similar these concepts are to the concepts of the same name in the SUSANNE corpus, since the precise definition of these terms as used in LDB is corpus-dependent.

powerful than SUSIE: e.g. a combination of different types of tags in a search is possible and easy to formulate.

A query or exploration scheme consists of two parts: the pattern and the activities. The activities can be initiated at the beginning of the search in the corpus, in a tree, after a match, after leaving a tree or after finishing a search. They are control structures guiding the search, output commands, or small programs (for statistical data, table construction, ...). The patterns consist of labels (slot values) and expressions (calculations and constraints put upon tree patterns). There is a match tree editor which allows for a quite intuitive (declarative) construction process of patterns. E.g. a (finite) sentence (`CAT = 'SF'`) with subject (`FUN = 'SU'`), verb (`FUN = 'VB'`) and object(`FUN = 'OD'`) looks like this in an LDB tree pattern (the `MATCH EFFECT` is an activity to be performed after a match has been found. In this case, control is handed back to the user):

```
PATTERN:

.---------------.
|CAT = 'SF'     |
'---------------'
  |
  |  .---------------.
  |--|FUN = 'SU'     |
  |  '---------------'
  |
  |  .---------------.
  |--|FUN = 'VB'     |
  |  '---------------'
  |
  |  .---------------.
  `--|FUN = 'OD'     |
     '---------------'

MATCH EFFECT:
    USER
```

An example of a result:

```
CORPUS drama DERIVED FROM ccppdr, TREE 1, LOCATION: 0602515, AMB:1

        .-1- IT
    .-1---2- LEAKS,
 -*-+-2- . . AND
    |   .-1- YOU
    '-3-+-2- KNOW
        '-3- IT.

  *       UTTERANCE:FINITE SENTENCE()
  *1          CONJOIN:FINITE SENTENCE()
  *11             SUBJECT:PRONOUN() : IT
  *12             VERB:VERB FINITE INTRANSITIVE() : LEAKS,
```

```
*2          COORDINATOR:COORDINATOR() : AND
*3          CONJOIN:FINITE SENTENCE()
*31             SUBJECT:PRONOUN() : YOU
*32             VERB:VERB FINITE MONOTRANSITIVE() : KNOW
*33             DIRECT OBJECT:PRONOUN() : IT.
```

The first line is the reference (corpus, tree, location, ambiguity), the second part is the analysis tree as shown to the user, the third part is again the tree, but in a different representation and with all information available provided. The reader can recognise the three slots:

```
<function>:<category>(<attributes>)
```

The numbers in front of the information indicate the node (by the path from the root * to the node). The match pattern can easily be retrieved in the result.

Van Halteren gives an overview of the possibilities: creating of patterns to match (in which the match can be a relative or an absolute place in a tree, a category, an attribute, a function (labels) or even a value assigned to variables (expressions)), and batch processing. One also can, using the variables, specify relations between nodes in a tree. E.g.

```
PATTERN:

.---------------.
|               |
'---------------'
  |
  |  .----------------.
  |--|CAT = "category |
  |  '----------------'
  |
  |  .----------------.
  `--|CAT = "category |
     '----------------'
```

meaning: look for a node which has (at least) two daughters, which have the same category `"category`. The use of variables also allows for a simple kind of statistical processing. An explicit aim of the developers is that the tool be easy to use. LDB allows the inspection of *all* matches in a tree, and does not (unless specified) abandon the search in a tree after the first match has been found (unlike SUSIE in the version we tested).

Unlike in SUSANNE, where sentences are assigned a unique parse, ambiguity is preserved, and it is up to the users to decide on the right analysis. One can request all possible parses of a sentence (cf. in the examples the indication `AMB:`). It is even possible to search for ambiguous sentences as such.

B Results

To demonstrate the facilities of the tool and the difference in tagging in the corpora, the queries executed in SUSIE are repeated here.

- Plural nouns cannot be retrieved if they are not parsed using attributes (e.g. attribute "num").
- Indirect object

```
PATTERN:

.---------------.
|FUN = 'OI'     |
'---------------'

MATCH EFFECT:
    USER
```

- Determiner (DE) and noun (NO):

```
PATTERN:

.---------------.
|               |
'---------------'
  |
  |  .---------------.
  |--|CAT = 'DE'     |
  |  '---------------'
  |
  |  .---------------.
  `--|CAT = 'NO'     |
     '---------------'

MATCH EFFECT:
    USER

CORPUS drama DERIVED FROM ccppdr, TREE 5, LOCATION: 0602517, AMB:1

    .-1- . . AND
 -*-+-2---1- THE
     |   '-2- GAS
    '-3- . . !

  *       UTTERANCE:NO CATEGORY()
  *1          IMMEDIATE CONSTITUENT:ADVERB() : AND
  *2          IMMEDIATE CONSTITUENT:NOUN PHRASE()
  *21             DETERMINER:DETERMINER() : THE
  *22             HEAD:NOUN() : GAS
  *3          PUNCTUATION MARK:EXCLAMATION MARK() : !
```

- A relative clause can be defined as finite sentence (SF) with as function POM, postmodifier:

```
SCHEME: RELATIVE CLAUSE

PATTERN:
```

```
.---------------.
|CAT = 'SF'     |
|FUN = 'POM'    |
'---------------'

MATCH EFFECT:
    USER
```

- There appears to be no simple way to formulate a query for reduced relative clauses.

3.3.3 Conclusions

Syntactically analysed corpora can overcome some of the problems that tagged corpora present. The identification of some syntactic structures is facilitated through the use of syntactic labels. Nevertheless the following problems remain:

- Identification
  Not all the syntactic categories that one is interested in using may have syntactic labels. One is therefore obliged to search for them using PoS tags as before.

  It may also be that syntactic attribute information (such as subcategorisation) is missing. Note that syntactic function information may help compensate for this lack. For verb subcategorisation, for example, if one wishes to distinguish between different types of verb that subcategorise for a NP and a PP:
  e.g. *he gives the book to Mary*
  he considers Mary as a friend
  one can make reference to the syntactic function of the PP constituent (prepositional object and complement of the object respectively).

- Syntactic information not sufficient?
  The SUSANNE corpus, as we saw, contains quite a lot of semantic information on its nouns. Semantic information is often necessary for test suite construction. For example, in subject verb agreement, the verb following a collective noun can be either singular or plural, although it itself is singular (e.g. *The committee has/have made the decision*). It remains an open question just how much more semantic information is needed in corpora for test suite construction.

- Ease of searching
  One must be careful, especially in the formulation of regular expression searches, that one neither fails to find targetted expressions, or returns false positives. It seems that manual inspection of the output is unavoidable.

4 Conclusions

We conclude by summarising some of the advantages and disadvantages of extracting test suites from corpora.

The main advantage of extracting test suites from corpora is that the data that one extracts constitute actually occurring phenomena, and are not contrived in any way. This is particularly important with respect to ill-formed data. It is difficult to predict what types of errors authors make when writing text—as discussed in Balkan et al. ([1]) error types vary according to level of education, whether the author is a native speaker, and if not, what their native language is. In Balkan *et al.* ([1]) we made the distinction between *competence* errors and *performance* errors. *Competence* errors are related to one's knowledge of a language, while *performance* are errors that occur as a result of text manipulation, treatment of formatted structures, punctuation, etc. The latter type of error is particularly difficult to predict, without the availability of error corpora.

Corpora also provide useful sources for extralinguistic phenomena, which is not always easily obtainable from other sources (e.g. grammar books).

Another advantage of using corpora as a source for test suite construction, is that the phenomena and lexical choices one finds are specific to a sublanguage, a fact that may be important if one is interested in knowing how a particular system performs on a particular text type or sublanguage. One can then tailor one's system (and test material) to just those phenomena and vocabulary.

However, we maintain that it is not desirable to simply lift test sentences from a corpus to form a test suite, for the following reasons:

- Complexity of constructions
  Even where search tools are able to identify a phenomenon, that phenomenon will usually cooccur with a variety of other phenomena. In order to test how the rules of a system interact it is important to be able to isolate the phenomenon from the other cooccurring phenomena. Thus additional, simpler, test items will always be need to be written to complement the complex examples.

  Dauphin and Lux ([6]) give, as an example of a complex sentence which is typical of their corpora, the following sentence, which shows different levels of coordination:
  Le but des opérations de vérification et d'entretien d'un appareil stocké longue duré est de connaître l'état de l'appareil et de la tenue du stockage afin de reconditionner l'appareil pour une nouvelle durée de stockage.

- Systematicity:
  Just as it is hard to find simple examples of phenomena in a corpus, so it is hard to find all the systematic variations of a phenomenon (e.g. examples of all the different verb subcategorisation frames), or

systematic ill-formed data or combinations of data. This is less of a problem if one only wants to write test material for a specific corpus (like Dauphin) but it is a problem if one wants to write general, reusable test material.

- Not enough information on test items
  As we noted above in the conclusions to section 3, it is an open question how much syntactic and semantic (and other) information requires to be in a corpus to allow the successful extraction of test suites. The situation is likely to improve, however, since there is currently a great deal of interest in corpus annotation schemes (see for example the SEMEVAL initiative, a semantic annotation scheme based on predicate argument structure proposed by Grishman ([17]), and it is likely that corpora will become better sources for the test suite writer as richer corpus annotation schemes are developed.

In the meantime we conclude that the main uses of corpora at present are:

- Providing an initial starting point for test suite construction (in the manner described by Lepage and Dauphin above)
- Testing coverage of an existing test suite
- Instantiating examples of an existing test suite at either the lexical or phrasal level
- Providing frequency information for weightings
- Providing output specifications
  for certain applications (e.g. syntactic analyses for parser evaluation).[10]

With this in mind, the TSNLP annotation scheme is designed to maximise these uses with the following information:

- the "ti-tag" annotation is designed to enable users to compare test suite patterns with corpora patterns for the purpose of establishing weightings or producing partial test suites that represent frequently occurring or highly weighted phenomena or structures in specific corpora or text types. No tag set is however provided by the TSNLP project.
- "ti-analysis" provides a slot for a constituent's category and function. These slots, while hand-parsed at present, may ultimately be lifted from a corpus, and can serve for the evaluation of parsers.

[10] Evaluations using hand-produced ideal parses of sentences from the Penn Treebank to compare parser output against are described in Harrison *et al.* [10], and Wojcik *et al.* [21]

- "ti-parameter" provides other types of linguistic information (e.g. subcategorisation frames) that we see as essential to the writing of systematic test suites. We suggest that information of this kind would be needed in corpus annotations to make them more usable for the extraction of test suites.

References

[1] Balkan (L.) – Meijer (S.) – Falkedal (K.) *et al.*, *Test Suite Design — Guidelines and Methodology*, Report to LRE 62-089 (D-WP2.1a), University of Essex, 1994.

[2] Balkan (L.) – Meijer (S.) – Falkedal (K.) *et al.*, *Issues in Test Suite Design*, Report to LRE 62-089 (D-WP2.1b), University of Essex, 1994.

[3] Brent (M. R.), *Automatic Acquisition of Subcategorization Frames from Untagged Text*, Proceedings of ACL 1991.

[4] Corley (M.) – Corley (S.), SUSIE corpus searching toolkit. Contact: `M.M.B.Corley@ex.ac.uk`.

[5] Dauphin (E.), *Etude de Corpus: un préalable pour l'adaptation des systèmes des Traduction Automatique aux besoins des utilisateurs.*

[6] Dauphin (E.) – Lux (V.), *Some difficulties in corpora studies*, input paper for TSNLP WP5.2.

[7] Estival (D.) – Falkedal (K.) *et al.*, *Analysis of Existing Test Suites*, Report to LRE 62-089 (D-WP1), University of Essex, 1994.

[8] Estival (D.) – Falkedal (K.) *et al.*, *Test Suite Design — Annotation Scheme*, Report to LRE 62-089 (D-WP2.2), University of Essex, 1994.

[9] Garside (R.) – Leech (G.) – Sampson, (G.), *The computational analysis of English. A corpus-based approach.* London, 1987.

[10] Harrison (P.) – Abney (s.) – Black (E.) *et al.*, *Evaluating Syntax Performance of Parser/Grammars of English*, in: Neal (J. G.) – Walter, (S. M.), *Natural Language Processing Systems Evaluation Workshop*, Calspan-UB Research Center, 1991.

[11] Leech (G.) – Barnett (R.) – Kahrel (P.), *SASG/2. Syntactic Annotation. Survey of Annotation Practices*, *s.l.*, 1995.

[12] Leech (G.) – Barnett (R.) – Kahrel (P.) *et al.*, *Detailed Overview of Existing Annotation Practices. SASG3*, *s.l.*, 1995.

[13] Lehrberger (J.) – Bourbeau (L.), *Machine Translation: Linguistic characteristics of MT systems and general methodology of evaluation*, John Betjamins, 1988.

[14] Lepage (Y.), *Test Suites as a means for the Evaluation of machine translation systems*, in: Falkedal (K.), *Proceedings of the Evaluators Forum*, Les Rasses, Vaud, 1991.

[15] Marcus, (M.) – Santorini, (B.) – Marcinkiewicz (M. A.), *Building a large annotated corpus of English: the Penn Treebank Project.*, in: *Computational Linguistics,* 1993, pp.313–30.

[16] Regnier (S.) – Dauphin (E.), *Test Suite Design at Aerospatiale*, input paper for TSNLP WP2.

[17] Grishman (R.), *Coding Predicate-Argument Structure*, document distributed by Elsnet as part of a text coding exercise.

[18] Sampson (G. R.) (forthcoming). *English for the Computer. The Susanne Corpus and Analytic Scheme*, Oxford, 1995.

[19] Sampson, (G.), *SUSANNE: A Domesday Book of English grammar*, in: Oostdijk, (N.) – de Haan, (P.) (*edd.*), *Corpus-based research into language*, *s.a.*, pp.169–187.

[20] van Halteren (H.) – van den Heuvel (T.), *Linguistic Exploitation of Syntactic Databases. The use of the Nijmegen* LDB *program*, (*Language and Computers: Studies in Practical Linguistics*, 5), Amsterdam – Atlanta(GA), 1990.

[21] Wojcik (R.) – Harrison (P.) – Bremer (J.), *Using bracketed parses to evaluate a grammar checking application*, ACL, 1993.

Appendix 1: Form tags and function tags in SUSANNE

Phrase level tags:

- N noun phrase
- V verb group
- J adjective phrase
- R adverb phrase
- P prepositional phrase
- D determiner phrase
- M numeral phrase
- G genitive phrase

The phrase subcategories are:

- Vo operator section of verb group, when separated from remainder of verb group, e.g. by subject-auxiliary inversion
- Vr remainder of verb group from which operator has been separated
- Vm verb group beginning with *am*
- Va verb group beginning with *are*
- Vs verb group beginning with *was*
- Vz verb group beginning with other 3rd-sing verb
- Vw verb group beginning with *were*

- Vj verb group beginning with *be*
- Vd verb group beginning with past tense
- Vi infinitival verb group
- Vg verb group beginning with present participle
- Vn verb group beginning with past participle
- Vc verb group beginning with modal
- Vk verb group containing emphatic DO
- Ve negative verb group
- Vf perfective verb group
- Vu progressive verb group
- Vp passive verb group
- Vb verb group ending with BE
- Vx verb group lacking main verb
- Vt catenative verb group

- Nq *wh*-noun phrase
- Nv *wh..* ever noun phrase
- Ne *I/me* head
- Ny *you* head
- Ni *it* head
- Nj adjective head
- Nn proper name
- Nu unit noun head
- Na marked as subject
- No marked as nonsubject
- Ns singular noun phrase
- Np plural noun phrase

- Jq *wh-* adjective phrase
- Jv *wh..ever* adjective phrase
- Jx measured absolute adjective phrase
- Jr measured comparative adjective phrase
- Jh postmodified adjective phrase

- Rq *wh-* adverb phrase
- Rv *wh..ever* adverb phrase

- Rx measured absolute adverb phrase
- Rr measured comparative adverb phrase
- Rs adverb conducive to asyndeton
- Rw quasi-nominal adverb

- Ps *of* phrase
- Pb *by* phrase
- Pq *wh* prepositional phrase
- Pv *wh..ever* prepositional phrase

- Dq *wh-* determiner phrase
- *wh..ever* determiner phrase
- singular determiner phrase
- Dp plural determiner phrase

- Ms phrase headed by *one*

Clauselevel tags include:

- S main clause
- Ss quoting clause embedded within quotation
- Fa adverbial clause
- Fn nominal clause
- Fr relative clause
- Ff fused relative
- Fc comparative clause
- Tg present participle clause
- Ti infinitival clause
- Tn past participle clause
- Tf for-to clause
- Tb bare nonfinite clause
- Tq infinitival relative clause
- Z reduced relative
- L other verbless clause
- A special *as* clause
- W *with* clause
- ? interrogative clause

- * imperative clause
- % subjunctive clause
- ! exclamatory clause or other item
- " vocative item

Root level tags include:

- O paragraph
- Oh heading
- Ot title (e.g. of a book)
- I interpolation
- Iq tag question
- Iu scientific citation

Complement functiontags:

- s logical subject
- o logical direct object
- S surface (and not logical) subject
- O surface (and not logical) direct object
- i indirect object
- u prepositional object
- e predicate complement of subject
- j predicate complement of object
- a agent of passive
- n particle of phrasal verb
- z complement of catenative
- x relative clause having higher clause as antecedent
- G "guest" having no grammatical role within its tagma

Adjunct funtiontags:

- p place
- q direction
- t time
- h manner or degree
- m modality
- c contingency
- r respect
- w comitative
- k benefactive
- b absolute

Appendix 2: Example grammar for SUSIE

```
n --> NP1c        % country name
n --> NP1p        % province name
n --> NP1t        % town name

% a few determiners from wordtagset
det --> AT        % the
det --> AT1       % a, an
det --> DD1a      % that (determiner, demonstrative pronoun
                  %       or qualifier)
det --> DD2a      % those
det --> DD2i      % these

np1 --> det n     % determiner + noun
np2 --> AT n      % 'the' + n
```

Appendix 3: View of a plain file from SUSANNE

This is what SUSANNE corpus files look like

```
A01:0010a       -       YB      <minbrk>        -       [Oh.Oh]
A01:0010b       -       AT      The     the     [O[S[Nns:s.
A01:0010c       -       NP1s    Fulton  Fulton  [Nns.
A01:0010d       -       NNL1cb  County  county  .Nns]
A01:0010e       -       JJ      Grand   grand   .
A01:0010f       -       NN1c    Jury    jury    .Nns:s]
A01:0010g       -       VVDv    said    say     [Vd.Vd]
A01:0010h       -       NPD1    Friday  Friday  [Nns:t.Nns:t]
A01:0010i       -       AT1     an      an      [Fn:o[Ns:s.
A01:0010j       -       NN1n    investigation   investigation   .
A01:0020a       -       IO      of      of      [Po.
A01:0020b       -       NP1t    Atlanta Atlanta [Ns[G[Nns.Nns]
A01:0020c       -       GG      +<apos>s        -       .G]
A01:0020d       -       JJ      recent  recent  .
A01:0020e       -       JJ      primary primary .
A01:0020f       -       NN1n    election        election        .Ns]Po]Ns:s]
A01:0020g       -       VVDv    produced        produce [Vd.Vd]
A01:0020h       -       YIL     <ldquo> -       .
A01:0020i       -       ATn     +no     no      [Ns:o.
A01:0020j       -       NN1u    evidence        evidence        .
A01:0020k       -       YIR     +<rdquo>        -       .
A01:0020m       -       CST     that    that    [Fn.
A01:0030a       -       DDy     any     any     [Np:s.
A01:0030b       -       NN2     irregularities  irregularity    .Np:s]
A01:0030c       -       VVDv    took    take    [Vd.Vd]
A01:0030d       -       NNL1c   place   place   [Ns:o.Ns:o]Fn]Ns:o]Fn:o]S]
A01:0030e       -       YF      +.      -       .O]
```

for *The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced "no evidence" that any irregularities took place.*

Appendix 4: Tags in the Nijmegen Corpus

```
FUNCTIONS:
        ????:UNIDENTIFIED
           A:LEFT APPOSITIVE
          AD:ADVERBIAL
          AS:AS
           B:APPOSITIVE
         CAS:PREPOSITIONAL COMPLEMENT OF AS
        CLIT:IT IN CLEFT SENTENCE
        CLOS:CLOSING QUOTE
         CLS:CLAUSE
        CLTL:CLEFT SENTENCE TAIL
          CO:OBJECT COMPLEMENT
        COOR:COORDINATOR
          CP:PREPOSITIONAL COMPLEMENT
          CS:SUBJECT COMPLEMENT
          CV:VERB COMPLEMENT
         DET:DETERMINER
        DIRN:STAGE DIRECTION
        DUMB:NO FUNCTION
          EF:FOCUSED ELEMENT
         ELL:ELLIPTIC
        FLAP:FLOATING APPOSITIVE
          HD:HEAD
          IC:IMMEDIATE CONSTITUENT
         INS:INSERT
        JOIN:CONJOIN
          LC:LOOSE COMPLEMENT
         LIM:LIMITER
        MARK:PUNCTUATION MARK
         MOD:MODIFIER
        NOOD:NOTIONAL OBJECT
        NOSU:NOTIONAL SUBJECT
          OD:DIRECT OBJECT
          OI:INDIRECT OBJECT
        OPEN:OPENING QUOTE
         POM:POSTMODIFIER
        PREM:PREMODIFIER
        PROD:PROVISIONAL OBJECT
         PRP:PREPOSITION
        PRSU:PROVISIONAL SUBJECT
          RP:REPORTED PART
          SU:SUBJECT
         SUB:SUBORDINATOR
          TO:INFINITIVE MARKER
        UTRP:REPORTED UTTERANCE
         UTT:UTTERANCE
          VB:VERB
         VIC:IMMED CONST VERBLESS CLAUSE
         VOC:VOCATIVE
          WC:PART OF COMPLEX WORD
           X:RIGHT APPOSITIVE MODIFIER
          XB:MODIFIED RIGHT APPOSITIVE

CATEGORIES:
        ????:UNIDENTIFIED
```

```
  AJ:ADJECTIVE
 AJP:ADJECTIVE PHRASE
  AN:ADJECTIVAL NUMERAL
  AS:AS
 ASP:AS PHRASE
  AV:ADVERB
 AVP:ADVERB PHRASE
CLIT:CLEFT IT
CLOS:CLOSING QUOTE
COOP:COORDINATOR PHRASE
COOR:COORDINATOR
  DE:DETERMINER
 DEP:DETERMINER PHRASE
 ELL:ELLIPTIC PHRASE
ENAV:ENOUGH-TYPE ADVERB
ENDE:ENOUGH-TYPE DETERMINER
  ER:COMPARATIVE
 ETH:EXISTENTIAL THERE
EXCA:EXTENDED CATEGORY
EXCL:EXCLAMATION MARK
EXVB:EXTENDED VERBAL
 GNP:GENITIVE NOUN PHRASE
  IN:INTERJECTION
 NAJ:NOMINAL ADJECTIVE
  NG:GENITIVE NOUN
  NN:NOMINAL NUMERAL
  NO:NOUN
NOCA:NO CATEGORY
  NP:NOUN PHRASE
OPEN:OPENING QUOTE
OTHR:OTHER
PAUS:PAUSE
  PC:PARENTHETIC CLAUSE
 PCL:PARENTHETIC CLAUSE LIMITER
 PIT:PROVISIONAL IT
  PN:PRONOUN
  PP:PREP PHRASE
PREP:PREPOSITION
  QU:QUESTION MARK
  RP:REPORTED PART
 REP:RELATER PHRASE
  SB:SUBCLAUSE
  SF:FINITE SENTENCE
  SN:NON-FINITE SENTENCE
 SUB:SUBORDINATOR
SUBP:SUBORDINATOR PHRASE
  TO:INFINITIVE MARKER
VAMF:VERB FINITE MODAL
VAPD:VERB PASTPART PRIMARY
VAPF:VERB FINITE PRIMARY
VAPG:VERB PRESPART PRIMARY
VAPI:VERB INFINITIVE PRIMARY
VAPM:VERB IMPERATIVE PRIMARY
VASD:VERB PASTPART SEMI-AUX
VASF:VERB FINITE SEMI-AUX
VASG:VERB PRESPART SEMI-AUX
VASI:VERB INFINITIVE SEMI-AUX
```

```
    VASM:VERB IMPERATIVE SEMI-AUX
      VB:VERB
    VBKD:VERB PASTPART COMPLEX
    VBKF:VERB FINITE COMPLEX
    VBKG:VERB PRESPART COMPLEX
    VBKI:VERB INFINITIVE COMPLEX
    VBKM:VERB IMPERATIVE COMPLEX
      VC:VERBLESS CLAUSE
     VCL:VERBLESS CLAUSE LIMITER
    VCOD:VERB PASTPART COPULA
    VCOF:VERB FINITE COPULA
    VCOG:VERB PRESPART COPULA
    VCOI:VERB INFINITIVE COPULA
    VCOM:VERB IMPERATIVE COPULA
    VITD:VERB PASTPART INTRANSITIVE
    VITF:VERB FINITE INTRANSITIVE
    VITG:VERB PRESPART INTRANSITIVE
    VITI:VERB INFINITIVE INTRANSITIVE
    VITM:VERB IMPERATIVE INTRANSITIVE
    VTCD:VERB PASTPART COMPLEX-TRANSITIVE
    VTCF:VERB FINITE COMPLEX-TRANSITIVE
    VTCG:VERB PRESPART COMPLEX-TRANSITIVE
    VTCI:VERB INFINITIVE COMPLEX-TRANSITIVE
    VTCM:VERB IMPERATIVE COMPLEX-TRANSITIVE
    VTDD:VERB PASTPART DITRANSITIVE
    VTDF:VERB FINITE DITRANSITIVE
    VTDG:VERB PRESPART DITRANSITIVE
    VTDI:VERB INFINITIVE DITRANSITIVE
    VTDM:VERB IMPERATIVE DITRANSITIVE
    VTRD:VERB PASTPART MONOTRANSITIVE
    VTRF:VERB FINITE MONOTRANSITIVE
    VTRG:VERB PRESPART MONOTRANSITIVE
    VTRI:VERB INFINITIVE MONOTRANSITIVE
    VTRM:VERB IMPERATIVE MONOTRANSITIVE
      WC:PART OF COMPLEX WORD
    WHDE:WH-DETERMINER
      XB:RIGHT OF MODIFIED APPOSITIVE

NO ATTRIBUTES
```

Appendix 5: View of a plain Nijmegen file

This is what the LDB corpus files look like

```
0800104 C.[ 9500 IN 9102 THE 2103 DARK 4103 DAYS 3203 OF 9104 WINTER, 3101 \
0800104 THIS 5101 IS E201 THE 2103 KIND 3103 OF 9104 WIMBLEDON 9905 DAY 3102 \
0800104 ONE 7703 \
0800105 DREAMS A203 ABOUT. 8800 IT 0201 'S F201 ARRIVED- A801 THE 2102 \
0800105 PERFECT 4102 SETTING 3102 FOR 9103 FINALS 9904 DAY. 3100 A 2502 \
```

for *C.[In the dark days of winter, this is the kind of Wimbledon day one dreams about. It's arrived- the perfect setting for finals day. A* (...)