Creating a Systemic Functional Grammar Corpus from the Penn Treebank

Matthew Honnibal and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{mhonn, james}@it.usyd.edu.au

Abstract

The lack of a large annotated systemic functional grammar (SFG) corpus has posed a significant challenge for the development of the theory. Automating SFG annotation is challenging because the theory uses a minimal constituency model, allocating as much of the work as possible to a set of hierarchically organised features.

In this paper we show that despite the unorthodox organisation of SFG, adapting existing resources remains the most practical way to create an annotated corpus. We present and analyse SFGBank, an automated conversion of the Penn Treebank into systemic functional grammar. The corpus is comparable to those available for other linguistic theories, offering many opportunities for new research.

1 Introduction

Systemic functional grammar (Halliday and Matthiessen, 2004) aims to describe the set of meaningful choices a speaker makes when putting a thought into words. Each of these choices is seen as a resource for shaping the meaning in a particular way, and the selection will have a distinct grammatical outcome as well as a semantic implication. The choices are presented hierarchically, so that early selections restrict other choices. For instance, if a speaker chooses imperative mood for a clause, they cannot choose a tense. Each selection is linked to a syntactic expression rule. When imperative mood is selected, the subject of the clause is suppressed; when interrogative mood is selected, the order of the subject and first auxiliary are reversed.

Systemic grammars are very different from grammars influenced by the formalist tradition. Systemic analysis locates a constituent within a typology, and yields a set of features that describe its salient properties. These features have proven useful for research in applied linguistics, on topics such as stylistics, discourse analysis and translation. As a generative theory, systemic grammars are less effective. There have been a few attempts, such as those discussed by O'Donnell and Bateman (2005), but as yet a wide coverage systemic grammar that can be used for tractable parsing has not been developed.

The lack of a corpus and parser has limited research on systemic grammars, as corpus studies have been restricted to small samples of manually coded examples, or imprecise queries of unannotated data. The corpus we present, obtained by converting the Penn Treebank, addresses this issue. It also suggests a way to automatically code novel text, by converting the output of a parser for a different formalism. This would also allow the use of SFG features for NLP applications to be explored, and support current research using SFG for applied linguistics.

The conversion process relies on a set of manually coded rules. The first step of the process is to collect SFG clauses and their constituents from parses in the Penn Treebank. Each clause constituent is then assigned up to three function labels, for the three simultaneous semantic and pragmatic structures Halliday (1970) describes. Finally, the system features are calculated, using rules referring to the function labels assigned in the previous step. This paper extends the work described in Honnibal (2004).

Proceedings of the 5th Workshop on Important Unresolved Matters, pages 89–96,
Ann Arbor, June 2005.

2 Related Work

Converting the Penn Treebank is the standard approach to creating a corpus annotated according to a specific linguistic theory. This has been the method used to create LTAG (Frank, 2001), LFG (Frank et al., 2003) and CCG (Hockenmaier and Steedman, 2005) corpora, among others. We employ a similar methodology, converting the corpus using manually specified rules.

Since the SFG annotation is semantically oriented, the work also bears some resemblance to Propbank (Palmer et al., 2005). However, Propbank is concerned with manually adding information to the Penn Treebank, rather than automatically reinterpreting the same information through the lens of a different linguistic theory.

We chose not to base our conversion on the Propbank annotation, as it does not currently cover the Brown or Switchboard sections of the Treebank. The wider variety of genres provided by these sections makes the corpus much more useful for SFG, since the theory devotes significant attention to pragmatic phenomena and stylistic variation.

3 Systemic Functional Grammar

Generating a constituent using a systemic functional grammar involves traversing a decision-tree-like structure referred to as a *system network*. The nodes of this tree are referred to as *systems*, and the options from the systems are referred to as *features*. At each system, the feature selected may add constraints on the type, number or order of the internal structure of the constituent. When the entire network has been traversed, the constraints are unified, and the required constituents generated.

In order to annotate a sentence according to a systemic functional grammar, we must specify the set of features encountered as the system network is traversed, and apply function labels to each constituent. The function labeling is required because the constraints are always specified according to the child constituents' function, rather than their form.

Constituents may have more than one function label, as SFG describes three *metafunctions*, following Halliday's (1969) argument that a clause is structured simultaneously as a communicative act, a piece of information, and a representation of reality.

Interpersonal function labels are assigned to clause constituents in determining the clause's communicative status. The most important interpersonal functions are *Subject* and *Finite*, since the relative position of the constituents bearing these labels largely determines whether the clause will be a question, statement or command.

The *textual* structure of the clause includes the functions *Theme* and *Rheme*, following Halliday's (1970) theory of information structure.

Finally, the *experiential* function of a constituent is its semantic role, described in terms of a small set of labels that are only minimally sensitive to the semantics of the predicate.

4 Annotation Implemented

We base our annotation on the clause network in the Nigel grammar (Mann and Matthiessen, 1983), as it is freely available and discussed at length in Matthiessen (1995). It is difficult to include annotation from the group and phrase networks, because of the flat bracketing of constituents in the Penn Treebank. The converted corpus has full coverage over all sections of the Penn Treebank 3 corpus.

We implement features from 41 systems from the clause network, out of a possible 62. The most prominent missing features relate to process type. The process type system classifies clauses as one of four broad semantic types: material, mental, verbal or relational, with subsequent systems making finer grained distinctions. This is mostly determined by the argument structure of the verb, but also depends on its lexical semantics. Process type assignment therefore suffers from word sense ambiguity, so we are unable to select from this system or others which depend on its result. Figure 1 gives an example of a clause with interpersonal, textual and experiential function labels applied to its constituents.

5 Creating the Corpus

SFG specifies the structure of a clause from 'above', by setting constraints that are imposed by the set of features selected from the system network. These constraints describe the structure in terms of interpersonal, textual and experiential function labels. These functions then determine the boundaries of the clause, by specifying its constituents.

Constituent	Interpersonal	Textual	Ideational
and	–	Txt. Theme	–
last year	Adjunct	Top. Theme	Circumstance
prices	Subject	Rheme	Participant
were	Finite	Rheme	–
quickly	Adjunct	Rheme	Circumstance
plummeting	Predicator	Rheme	Process

Table 1: SFG function labels assigned to clause constituents.

```
preprocess(parse)
clauses = []
for word in parse.words():
  if isPredicate(word):
    constituents = getConstituents(word)
      clauses.append(constituents)
```

Figure 2: Conversion algorithm.

The Penn Treebank provides rich syntactic trees, specifying the structure of the sentence. We therefore proceed from 'below', using the Penn Treebank to find clauses and their constituents, then applying function labels to them, and using the function labels as the basis for rules to traverse the system network.

5.1 Finding Constituents

In this stage, we search the Treebank parse for SFG clauses, and collect their constituents. Clauses are identified by searching for predicates that head them, and constituents are collecting by traversing upwards from the predicate, collecting the nodes' siblings until we hit an S node.

There are a few common constructions which present problems for one or both of these procedures. These exceptions are handled by preprocessing the Treebank tree, changing its structure to be compatible with the predicate and constituent extraction algorithms. Figure 2 describes the conversion process more formally.

5.1.1 Finding predicates

A predicate is the main verb in the clause. In the Treebank annotation, the predicate will be the word attached to the lowest node in a VP chain, because auxiliaries attach higher up. Figure 3 describes the function to decide whether a word is a predicate. Essentially, we want words that are the last word attached to a VP, that do not have a VP sibling.

Figure 1 marks the predicates and constituents in a Treebank parse. The predicates are underlined, and the constituents numbered to match the predicate.

```
if verb.parent.label == 'VP':
  for sibling in verb.parent.children:
    if sibling.isWord():
      if sibling.offset > verb.offset:
        return False
    if sibling.label == 'VP':
      return False
return True
```

Figure 3: Determining whether a word is a predicate.

```
node = predicate
constituents = [predicate]
while node.label not in clauseLabels:
  for sibling in node.parent.children:
    if sibling != node:
      constituents.append(sibling)
for sibling in node.parent.children:
  if sibling != node
  and sibling.label in conjOrWHLabels:
    constituents.append(sibling)
```

Figure 4: Finding constituents.

5.1.2 Getting Constituents

Once we have a predicate, we can traverse the tree around it to collect the constituents in the clause it heads. We do this by collecting its siblings and moving up the tree, collecting the 'uncle' nodes, until we hit the top of the clause. Figure 4 describes the process more formally. The final loop collects conjunctions and WH constituents that attach alongside the clause node, such as the 'which' in Figure 1.

5.1.3 Pre-processing Ellipsis and Gapping

Ellipsis and gapping involve two or more predicates sharing some constituents. When the sharing can be denoted using the tree structure, by placing the shared items above the point where the VPs fork, we refer to the construction as ellipsis. Figure 5 shows a sentence with a subject and an auxiliary shared between two predicates. 3.4% of predicates share at least one constituent with another clause via ellipsis. We pre-process ellipsis constructions by inserting an S node above each VP after the first, and adding traces for the shared constituents.

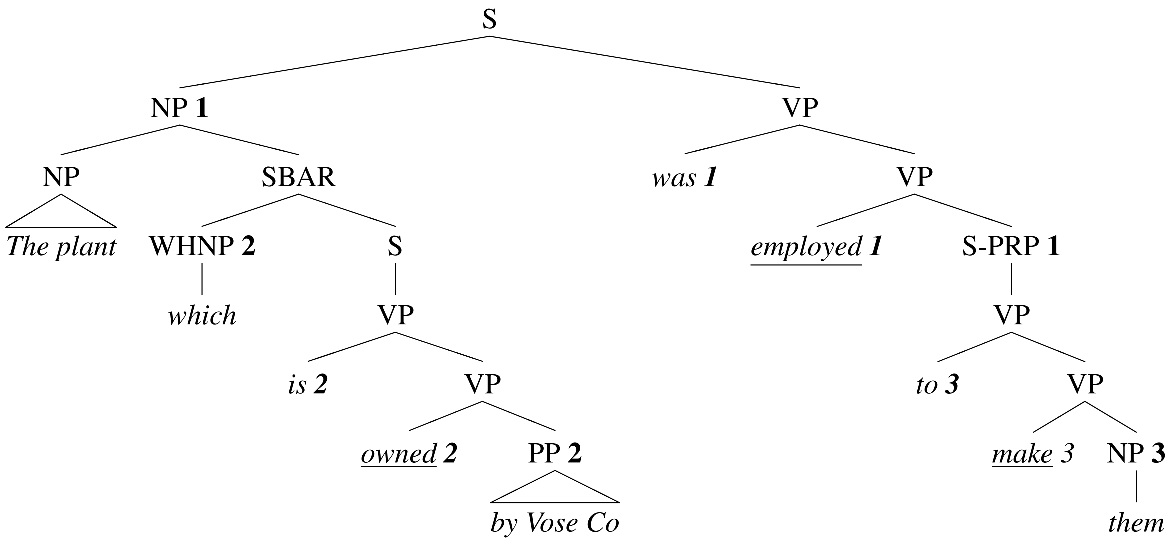

Figure 1: A parse tree with predicates underlined and constituents numbered.

In gapping constructions, the shared constituent is the predicate itself, and what differs between the two clauses are the arguments. The Treebank uses special trace rules to describe which arguments must be copied across to the gapped clause. We create traces to the shared constituents and add them to each gapped clause, so that the trace of the verb will be picked up as a predicate later on. Gapping is a very rare phenomenon – only 0.02% clauses have gapped predicates.

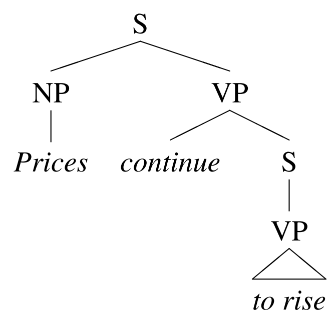

Figure 6: Treebank representation of a sentence with a semi-auxiliary.

5.1.4 Pre-processing Semi-auxiliaries

In Figure 6 the verb ‘continue’ will match our rules for predicate extraction, described in Section 5.1. SFG analyses this and other ‘semi-auxiliaries’ (Quirk et al., 1991) as a serial verb construction, rather than a matrix clause and a complement clause. Since we want to treat the finite verb as though it were an auxiliary, we pre-process these cases by simply deleting the S node, and attaching its children directly to the semi-auxiliary’s VP.

Defining the semi-auxiliary constructions is not so simple, however. Quirk et al. note that some of these verbs are more like auxiliaries than others, and organise them into a rough gradient according to their formal properties. The problem is that there is not clear agreement in the SFG literature about where the line should be drawn. Matthiessen (1995) describes all non-finite sentential complements as serial-verb constructions. Martin et al. (1997) argue that verbs such as ‘want’ impose selectional restrictions on the subject, and therefore should be treated as full verbs with a clause complement. Other compromises are possible as well.

Using Matthiessen’s definition, we collect 5.3% fewer predicates than if we treated all semi-auxiliaries as main verbs. If the complement clause has a different subject from the parent clause, when the two are merged the new verb will seem to have extra arguments. 58% of these mergings introduce an extra argument in this way. For example,

Investors want the market to boom

will be analysed as though *boom* has two arguments, *investors* and *market*. We prevent this from occurring by adding an extra condition for merging clauses, stating that the subject of the embedded clause should be a trace co-indexed with the subject of the parent clause.

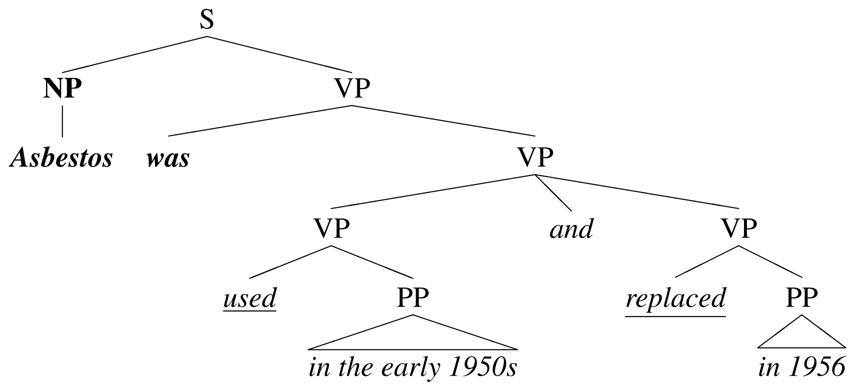

Figure 5: Treebank representation of ellipsis. Predicates are underlined, shared items are in bold.

5.2 Constituent functions

As discussed above, we attach up to three function labels to each clause constituent, one for each *metafunction*. The rules to do this rely on the order of constituents and the function dash-tags in the Penn Treebank. Some experiential function rules also refer to interpersonal labels, and some textual function rules refer to experiential labels.

5.2.1 Interpersonal Function Labels

The possible interpersonal function labels we assign are *Subject*, *Complement*, *Adjunct*, *Finite*, and *Predicator*. The Finite and Predicator are the first tensed verb, and the predicate respectively. If there are no auxiliary verbs, Halliday and Matthiessen (2004) describes the predicate as functioning both as Finite and Predicator. Since this is the only case in which a constituent would receive multiple labels from a single metafunction, we instead assign the single label *Finite/Predicator*.

For NPs, Subjects, Complements and Adjuncts are distinguished using the Penn Treebank's dash-tag function labels. SFG always assigns prepositional phrases the label Adjunct. All NP constituents that are not marked with an adverbial function tag in the Treebank are labeled *Complement*. Conjunctions are not assigned interpersonal functions.

5.2.2 Experiential Function Labels

The experiential function labels we assign are *Participant*, *Process* and *Circumstance*. This is a simplification of the function labels described by Halliday and Matthiessen (2004), as Participants are usually subdivided into what other linguistic theories refer to as semantic roles. SFG has its own semantic role description, which relies on *process type* features. For instance, Participants in a *verbal* process like 'say' have the role options *Sayer*, *Target*, *Receiver* and *Verbiage*.

Distinguishing process types requires a word sense disambiguated corpus and a word sense sensitive process type lexicon. While there is a significant intersection between the Penn Treebank and the Semcor word sense disambiguated corpus, there is currently no suitable process type lexicon. Consequently, Participants have not been subtyped. The Process is simply the verb phrase, while the Subject and Complements are Participants.

5.2.3 Textual Function labels

The textual metafunction describes the information structure of the clause. Halliday's textual function labels are *Textual Theme*, *Interpersonal Theme*, *Topical Theme* and *Rheme*. Theme and Rheme are often referred to as Topic and Comment in other theories of information structure (Vallduvi, 1993). Theories also disagree about exactly where to draw the boundary between the two.

In Halliday's theory, the Rheme begins after the first full constituent with an experiential function label, and extends to the end of the clause. The first constituent with an experiential function is labeled Topical Theme. Constituents before it are labeled either Interpersonal Theme or Textual Theme. Auxiliaries and vocatives are labeled Interpersonal Theme, while conjunctions are labeled Textual Theme.

System	Null %	Feature 1	Feature 2
clause class	0%	major (86%)	minor (13%)
agency	13%	effective (52%)	middle (34%)
conjunction	13%	non-conjuncted (64%)	conjuncted (21%)
finiteness	13%	finite (67%)	non-finite (19%)
polarity	13%	positive (81%)	negative (4%)
rank	13%	ranking (66%)	shifted (19%)
secondary/beta clause	13%	false (58%)	true (28%)
status	13%	bound (45%)	free (41%)
deicticity	32%	temporal (60%)	modal (7%)
person	32%	non-interactant (54%)	interactant (13%)
theme selection	32%	unmarked (58%)	marked (9%)
voice	47%	active (45%)	passive (6%)
embed type	80%	nominal qualifier (15%)	other qualifier (3%)
theme role	90%	as adjunct (7%)	as process (1%)
passive agency	93%	non-agentive (5%)	agentive (1%)

Table 2: Selected systems and how often their features are selected.

5.3 System Selections

As discussed above, the system features are organised into hierarchies, with every feature assuming a null value unless its system's entry condition is met. We therefore approach the system network much like a decision tree, using rules to control how the network is traversed.

The rules used to traverse the network cannot be explained here in full, as there are 41 such decision functions currently implemented. Table 2 lists a few of the systems we implement, along with how often their features are selected. Because the system network is organised hierarchically, a selection will not always be made from a given system, since the 'entry condition' may not be met. For instance, the feature agency=effective is an entry condition for the voice system, so if a clause is middle, no voice will be selected. The Null % column describes how often the entry condition of the clause is not met. Systems further down the heirarchy will obviously be relevant less often, as will systems which describe a finer grained distinction for an already rare feature.

The following sections describe the system network in terms of four general regions. The systems within each region largely sub-categorise each other, or relate to the same grammatical phenomenon.

5.4 Mood systems

Assuming the clause is independent, the major mood options are declarative, interrogative and imperative. Deciding between these is quite simple: in interrogative clauses, the Subject occurs after the first auxiliary. Imperative clauses have no Subject.

There are a few more granular systems for interrogative clauses, recording whether the question is polar or WH. If the clause is WH interrogative, there are two further features recording whether the requested constituent functions as Subject, Adjunct or Complement. The values of these features are quite simple to calculate, by finding the WH element among the constituents and retrieving its interpersonal function.

If the clause is not imperative, there are systems recording the person (first, second or third) of the subject, and whether the first auxiliary is modal, present tense, past tense, or future tense. SFG describes three tenses in English, regarding 'will' and 'shall' auxiliaries as future tense markers, rather than modals.

If the clause is imperative, there is a further system recording whether the clause is the 'jussive' imperative with 'let's', an 'oblative' imperative with 'let me', or a second person imperative. If the imperative is second person, a further feature records whether the 'you' is explicit or implied.

There are also features recording the 'polarity' of the clause: whether it is positive or negative, and, if negative, whether the negative marker is full-formed or cliticised as -n't.

5.5 Voice systems

In the Nigel grammar, the first voice distinction drawn is not between active and passive, but between transitive and intransitive clauses. Intransitive clauses cannot be passivised, as there is no Complement to shift to Subject. It therefore makes sense to carve these off first. If the clause is transitive, another system records whether it is active or passive. The rules to draw this distinction simply look at the verb phrase, checking whether the last auxiliary is a form of the verb 'be' and the lexical verb has a past participle part-of-speech tag. Finally, a further system records whether passive clauses have an agent introduced by 'by'.

5.6 Theme systems

Theme systems record what occurs at the start of the clause. Typically in English, the first major constituent will be the logical subject, and hence also the Topical Theme. A system records whether this is or is not the case. If the clause is finite and the logical subject is not the Topical Theme, the clause is said to have a 'marked' theme. Verb phrase Topical Themes are considered unmarked if the clause is imperative. A further system records whether the Topical Theme is the logical object (as in passivisation), or whether it is a fronted Adjunct. Passive clauses may have a fronted Adjunct, so does not necessarily have a logical object as Topical Theme. There are two further systems recording whether the clause has a Textual Theme and/or an Interpersonal Theme.

5.7 Taxis systems

Taxis systems record dependency relationships between clauses. There are two types of information: whether the attachment is made through coordination or subordination, and the semantic type of the attachment. Broadly, semantic type is between 'expansion' and 'projection', projection being reported (or quoted) speech or thought. A further system records the subtype of expansion clauses, which is quite a subtle distinction. Unfortunately Halliday chose thoroughly unhelpful terminology for this distinction: his subtypes of expansion are elaboration, enhancement and extension. Enhancing clauses are essentially adverbial, and are almost always subordinate. Extending clauses, by contrast, are approximately the 'and' relationship, and are almost always coordinate. Elaborating clauses qualify or further define the information in the clause they are attached to. Elaborating clauses can be either subordinate or coordinate. Subordinate elaborating clauses are non-defining relative clauses, while coordinate elaborating clauses are usually introduced by a conjunctive adjunct, like 'particularly'.

6 Accuracy

In order to evaluate the accuracy of the conversion process, we manually evaluated the constituency structure of a randomly selected sample of 200 clauses. The conversion heuristics were developed on section 00 of the Wall Street Journal and section 2 of Switchboard, while the evaluation sentences were sampled from the rest of the Penn Treebank.

We limited evaluation to the constituency conversion process, in order to examine more clauses. The function labels are calculated from the constituency conversion, while the system features are calculated from the function labels and other system features. Since the system network is like a decision tree, whether a feature is null-valued depends on prior feature decisions. These dependencies in the annotation mean that evaluating all of it involves some redundancy. We therefore evaluated the constituency structure, since it did not depend on any of the other annotation, and the conversion heuristics involved in calculating it were more complicated than those for the function labels and system features.

In the 200 clause sample, we found three clauses with faulty constituency structures. One of these was the result of a part-of-speech tag error in the Treebank. The other two errors were conjunctions that were incorrectly replicated in ellipsis clauses.

7 Conclusion

The Penn Treebank was designed as a largely theory neutral corpus. In deciding on an annotation scheme, it emphasised the need to have its annotators work quickly and consistent, rather than fidelity to any particular linguistic theory (Marcus et al., 1994).

The fact that it has been successfully converted to so many other annotation schemes suggests that its annotation is indeed consistent and fine grained. It is therefore unsurprising that it is possible to convert it to SFG as well. Despite historically different concerns, SFG still fundamentally agrees with other theories about which constructions are syntactically distinct — it simply has an unorthodox strategy for representing that variation, delegating more work to feature structures and less work to the syntactic representation.

Now that a sizable SFG corpus has been created, it can be put to use for linguistic and NLP research. Linguistically, we suggest that it would be interesting to use the corpus to explore some of the assertions in the literature that have until now been untestable. For instance, Halliday and Matthiessen (2004) suggests that the motivation for passivisation is largely structural — what comes first in a clause is an important choice in English. This implies that the combination of passive voice and a fronted adjunct should be unlikely. There should be many such simple queries that can shed interesting light on abstract claims in the literature.

Large annotated corpora are currently very important for parsing research, making this work a vital first step towards exploring whether SFG annotation can be automated. The fact that Treebank parses can be converted into SFG annotation suggests it can be, although most parsers do not replicate the dash-tags attached to Treebank labels, which are necessary to distinguish SFG categories in our conversion system.

Even without automating annotation, the corpus offers some potential for investigating how useful SFG features are for NLP tasks. The Penn Treebank includes texts from a variety of genres, including newspaper text, literature and spoken dialogue. The Switchboard section of the corpus also comes with various demographic properties about the speakers, and is over a million words. We therefore suggest that gold standard SFG features could be used in some simple document classification experiments, such as predicting the gender or education level of speakers in the Switchboard corpus.

8 Acknowledgments

We would like to thanks the anonymous reviewers for their helpful comments. James Curran was funded under ARC Discovery grants DP0453131 and DP0665973.

References

Anette Frank. 2001. Treebank conversion: Converting the NEGRA treebank to an LTAG grammar. In *Proceedings of the EUROLAN Workshop on Multi-layer Corpus-based Analysis*. Iasi, Romania.

Anette Frank, Louisa Sadler, Josef van Genabith, and Andy Way. 2003. *From Treebank Resources To LFG F-Structures - Automatic F-Structure Annotation of Treebank Trees and CFGs extracted from Treebanks*. Kluwer, Dordrecht.

Michael A. K. Halliday. 1969. Options and functions in the English clause. *Brno Studies in English*, 8:82–88. Reprinted in Halliday and Martin (eds.)(1981) Readings in Systemic Linguistics, Batsford, London.

Michael A. K. Halliday. 1970. Language structure and language function. In John Lyons, editor, *New Horizons in Linguistics*. Penguin, Harmondsworth.

Michael A. K. Halliday and Christian M. I. M. Matthiessen. 2004. *An Introduction to Functional Grammar*. Edward Arnold, London, third edition.

Julia Hockenmaier and Mark Steedman. 2005. Ccgbank manual. Technical Report MS-CIS-05-09, University of Pennsylvania.

Matthew Honnibal. 2004. Converting the Penn Treebank to Systemic Sunctional Grammar. In *Proceedings of the Australasian Language Technology Workshop (ALTW04)*.

William C. Mann and Christian M. I. M. Matthiessen. 1983. An overview of the Nigel text generation grammar. Technical Report RR-83-113, USC/Information Sciences Institute.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

James R. Martin, Christian M. I. M. Matthiessen, and Clare Painter. 1997. *Working with Functional Grammar*. Arnold, London.

Christian Matthiessen. 1995. *Lexicogrammatical Cartography*. International Language Sciences Publishers, Tokyo, Taipei and Dallas.

Michael O'Donnell and John A. Bateman. 2005. SFL in computational contexts: a contemporary history. In J. Webster, R. Hasan, and C. M. I. M. Matthiessen, editors, *Continuing Discourse on Language: A functional perspective*, pages 343–382. Equinox, London.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1991. *A Grammar of Contemporary English*. Longman, London.

Enric Vallduvi. 1993. Information packing: A survey. Technical Report HCRC/RP-44, Universiy of Edinburgh.