Systemic Functional Corpus Resources: Issues in Development and Deployment

Elke Teich, Richard Eckart, Monica Holtz

Darmstadt University of Technology
Institute of Linguistics and Literary Studies
Department of English Linguistics
Hochschulstrasse 1, 64289 Darmstadt, Germany
E-mail: {LASTNAME}@linglit.tu-darmstadt.de

1 Introduction

As the concern with linguistically interpreted corpora of authentic natural language texts is growing, the research activities in this field are becoming increasingly diversified – both in terms of the types of linguistic phenomena dealt with and in terms of the linguistic theories engaging in it.

Recently, linguistic phenomena other than syntactic ones, notably semantic and discourse phenomena, have received more attention. Here, existing resources are often taken as a backbone in an attempt to create an explicit link between syntactic and semantic annotation (e.g., PropBank [2] which takes the Penn Treebank [12] as a basis). Also, there is a growing number of linguistic theories that have come to acknowledge the need to work with authentic linguistic data including Dependency Grammar [5], LFG [10] and HPSG [16]. The motivations for this are theory testing, on the one hand, and building models for NLP applications (e.g., probabilistic parsers), on the other.

In this paper we discuss the development and deployment of corpus resources based on *Systemic Functional Linguistics* (SFL) [11]. SFL has a strong tradition in text analysis and interpretation. Its focus is on register variation, where a register is said to be characterized by the greater-than-random co-occurrence of particular linguistic features. However, with few exceptions (e.g., [13]) SFL has so far primarily worked with text samples instead of corpora, thus missing the chance of further developing its theory in the direction of probabilistic grammar. We therefore see a need on the part of SFL for high-quality corpus resources and tools that support the development and deployment of such resources.

Hajič, J. and Nivre, J. (eds.): Proceedings of the TLT 2006, pp. 247–258.

Our primary motivation in developing SFL-based corpus resources lies in this need. More concretely, we are currently building up an archive of English academic texts (articles from journals and conference proceedings) from selected scientific disciplines (including biology, mechanical engineering, computer science). The final size of this text archive will be around 10 million tokens. Portions of this archive will be compiled into corpora and lexico-grammatically annotated for purposes of register analysis, for example for investigating recent change in language (emerging registers), as well as for learning register-specific grammars. From the perspective of treebanks and linguistically interpreted corpora more generally, our goal is to contribute an additional angle on the analysis of linguistic data so as to further the common understanding of issues in linguistic data representation and deployment.

The paper is organized as follows. We start with a brief synopsis of the main features of SFL theory and present the particular requirements on SFL-based corpora (Section 2). We then describe our current practice in corpus annotation, also pointing out shortcomings of this practice (Section 3). Finally, we present our current solution to corpus representation and give an example of corpus deployment for linguistic analysis (Section 4). We conclude with a summary and issues for future work which include the development of tools for annotation, the refinement of the data model underlying corpus representation and the development of facilities for linguistic analysis (query, statistical analysis) as well as grammar modeling (Section 5).

2 Systemic Functional Linguistics (SFL)

2.1 SFL Model

SFL theory considers language a *stratified resource* with the strata of context, semantics, lexico-grammar and phonology (for spoken language) (cf. [11]). At the core is the lexico-grammatical stratum and this is also our main concern in corpus annotation. What is special about *Systemic Functional Grammar* (SFG) is its strong functional bias. This is formulated in terms of the concept of *metafunction*. There are three metafunctions: the *ideational* – expressing propositional types of linguistic information (e.g., predicate-argument structure); the *interpersonal* – expressing interactional and evaluative types of linguistic information (e.g., mood, modality, polarity); and the *textual* – expressing information about linguistic properties of discourse organization (e.g., theme-rheme, given-new).

An SFG is organized as a feature lattice[1] with features from all three meta-

[1] In this regard, SFG is quite similar to HPSG.

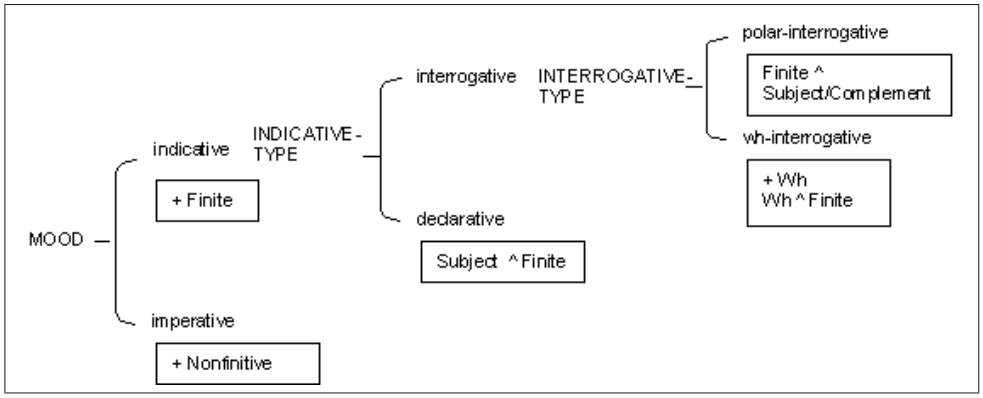

Figure 1: An SFG system network for English MOOD

Features	Process-type: mental; Mood: declarative; Theme: Marked				
Clause	*in the next step*	*the receiver*	*needs*	*to know*	*the details*
ideational	Time	Senser		Process	Phenomenon
inter-personal	Residue	Mood		Residue	
	Adjunct	Subject	Finite	Predicator	Complement
textual	Theme	Rheme			
	Topical				

Figure 2: Example of an SFG function structure (clause rank)

functions that hold simultaneously for a given grammatical unit (e.g., a clause or a phrase) and carry constraints on the grammatical structure of that unit. For example, a feature *declarative* will constrain the grammatical structure of a clause to include a Subject and a Finite element, where the Subject is ordered before the Finite (for English).

The grammatical feature lattice is called *system network* (see Figure 1 for an example). The system network represents the *paradigmatic* linguistic organization and constitutes the core of an SFG. Applying an SFG in analysis involves choosing the features that describe a given linguistic instance from the system network and spelling out the constraints on the grammatical structure associated with the selected features in the form of a *function structure*. The function structure represents *syntagmatic* linguistic organization and unifies information from the three metafunctions into one representation, which also includes the set of features selected. For an example see Figure 2. Similar analyses are carried out for lower ranks, notably groups/phrases and words.

2.2 Requirements on SFL-based Corpora

The concern with multiple dimensions of linguistic analysis in SFL poses particular demands on the representation, annotation and deployment of SFL-based corpora.

Traditional corpora (e.g., Brown, LOB or BNC) are typically annotated at the word rank only (word class and grammatical category encoded in the form of PoS tags). Treebanks go one step further in linguistic interpretation annotating phrase or dependency structure (e.g., [12, 5]). Neither traditional corpora nor treebanks were originally designed to include other aspects of linguistic description. This causes difficulties when additional types of annotations need to be added, as for example reported in [2] for the Propbank extension of PennTree in terms of predicate-argument structure.

Corpora annotated on the basis of SFL need to acknowledge more than one descriptional dimensions from the start. For grammatical annotation both the paradigmatic (system network) and the syntagmatic (function structure) aspects of linguistic organization and the metafunctions (ideational, interpersonal, textual) need to be covered. This has implications for the annotation as well as the representation and subsequent deployment of SFL-based corpus resources:

Annotation. Tools for annotation are needed that support the definition of annotation schemes in the form of fully explicit systemic functional grammars (system networks plus constraints on structure).[2] Also, assuming a human annotator, the annotation process needs to be supported to ensure correctness of application of the annotation schemes.

Representation. A function structure superimposes three different views on a given string. In many cases, these cannot be expressed by a single tree, since each may divide a given string differently, thus creating overlapping segments (see again the example in Figure 2). For representation, one thus needs to opt for a more expressive data model, as e.g., proposed by [4] or [17].

Deployment. For use in linguistic analysis, an SFL-based corpus has to be queryable according to the needs of the user. Adequate query techniques have to be defined and implemented (cf. [19]).

The following sections describe our current practice in building and working with SFL-based corpus resources and discuss the needs for extension and improvement of existing tools.

[2] This is comparable to grammar development support in parsing or generation (cf. the grammar development tools for SFGs included in the KPML generation system [3]).

3 Existing Tools

Most systemic functional annotation has to be done manually since automatic SFG analysis is beyond the capabilities of currently available tools [13]. Apart from PoS tagging and some basic phrase structure parsing which we carry out automatically using the TnT tagger [6] and LTChunk [9], annotations of the kinds described in Section 2 have to be done manually.

To support annotation and subsequent query, a number of tools are available, but none of them allows the specification of a fully-fledged SFG. The best known among these tools is Systemic Coder [15]. Systemic Coder provides support for defining annotation schemes in the form of system networks and guides the user through the annotation scheme in the process of annotation. However, the interpretation of function structure specifications in the system network definitions is not supported. Another limitation of Systemic Coder is that only one level of segmentation can be dealt with at a time, i.e., discontinuous segments (e.g., preposition stranding) as well as embedded units (e.g., relative clauses) present a problem. Once a corpus has been annotated, it can be queried for features or feature combinations and a concordance is returned. Also, it is possible to derive some simple statistics, again querying for the features. The annotated data are stored in XML.

A more recent tool, LBIS Coder [18], additionally allows to interpret constraints on structure. Also, it is more flexible concerning segmentation, covering e.g., embedded units. The corpus data are stored in XML. However, LBIS Coder does not offer the possibility of statistical analysis.

Finally, there is the SysFan tool [20]. SysFan can handle multiple annotations at a time and the annotator can specify structural constraints with the features. However, there is no support for checking the application of those constraints, which makes this functionality practically useless. The annotated corpus is stored as a relational database and can be queried accordingly. SysFan's commitment to a proprietary relational database makes it hard to export data to applications that follow a different philosophy, e.g., a semi-structured XML-based data design.

For the time being we live with these imperfections. We create feature annotations with Systemic Coder, export the XML and then add function structure annotations manually using a generic XML editor. Each annotation created is treated as an independent layer. To relate the different layers to each other we have developed our own tool, AnnoLab [7], which is based on a multi-layer data model. The problem of potentially conflicting segmentations at annotation time is thus avoided because annotations are kept separately and are related to each other only at query time. The AnnoLab tool and its underlying data model are described in the next section.

4 A Modular, Flexible Approach

We have examined a number of different approaches of dealing with multi-layer annotations and from those synthesized a modular data model [8]. This model divides the representation of annotations into four tiers:

- The *signal tier* contains the *signals*, instances of primary data, such as text or speech.
- The *structure tier* contains the structures that are superimposed on the primary data, e.g. a function structure or a parse tree. These structures are called *layers* and are made up of typed *elements*. Each layer provides a set of ways in which these elements can relate to each other, e.g. a tree-layer offers a parent-child relation and a sibling relation.
- The *feature tier* contains *features* associated with structure elements of the signal tier. Features carry a name and a value.
- The *location tier* allows mapping between the signal tier and the structure tier. The signals of the signal tier and the layers of the structure tier are connected in a stand-off fashion by means of *segments*. A segment describes a region of the primary data by means of anchors marking its boundaries.

Figure 3 illustrates how we apply this model to represent a multi-layer SFL annotation. It shows five annotation layers, one for each metafunction and one for each PoS and phrase structure. The layers are made up of annotation elements, such as *Clause*, *Subject* or *Process*, some of which carry features such as *Process-type: Mental* or *Part-of-Speech: Verb*.[3] The leaf elements are associated with one or more segments which connect the annotations to the text string.

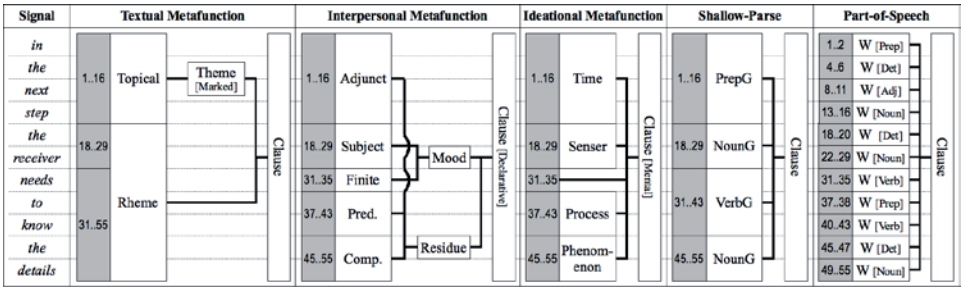

Figure 3: Multi-layer SFL, phrase-structure and PoS annotation

[3] Due to limited space only the feature value is shown in Figure 3.

In order to make use of a resource annotated on the basis of this data model, we have collected a set of query operators and associated them with the four tiers.

At the signal tier, query operators can be of two different kinds. *Location-based* operators take segments as parameters and return pieces of primary data addressed by these segments. *Content-based* operators match against the primary data and return segments addressing the matching sections.

At the structure and feature tiers, query operators come in the form of *axes* along which it is possible to navigate through the annotation layers. These axes are induced by the element-to-element relations of the layer. Some of the axes induced by a tree-layer are the *child/parent*, *descendant/ancestor*, *following/preceding sibling* and *following/preceding element* axes. One axis also allows to access the features associated with an element.

At the location tier, queries operate on segments. The query operators defined allow to test for *overlaps*, *adjacency*, *containment* or *alignment* as well as *union* and *intersection* of segments.

The central design aspect of the data model is to highlight opportunities for modularizing annotation processing systems. For instance, an annotation system that works for textual signals can in principle be easily extended to support audio signals: both are linear in nature and can be used in combination with the same location, structure and feature devices. Only that part of the system dealing with the signal aspects has to be modified. Similarly, when an annotation requires a particular annotation structure in the layers, only the implementation of the structure tier has to be adapted. The approach allows for implementations of support for different kinds of signals (text, audio, video), location mappings (intervals, areas, spaces), layer structures (tables, trees, graphs) and features (simple features or feature structures) to exist in parallel and be plugged together to suite a particular task.

The design of the data model and query operators was greatly influenced by the XML data model and the associated XPath and XQuery languages. The reason for using these as a foundation is their inherent suitability for the task of annotation, especially the annotation with trees.

Based on the conception of the modular data model we have implemented an XML-based workbench for multi-layer annotation called AnnoLab. While XML and XPath/XQuery are not immediately suitable for all our annotation needs, they are reasonably suitable for representing and querying individual annotation layers. To support multi-layer annotation, we have replaced XML's text nodes by segments, transforming them into a stand-off annotation. In this way, AnnoLab can treat any XML-based annotation containing the annotated text in the text-nodes, e.g. Systemic Coder files, as an annotation layer. Figure 4 gives an impression of the resulting XML code.

```
<layer name="Interpersonal Metafunction">
  <Clause Type="Declarative">
    <Residue>
      <Adjunct><segment start="1" end="16"/></Adjunct>
      <Predicator><segment start="37" end="43"/></Predicator>
      <Complement><segment start="45" end="55"/></Complement>
    </Residue>
    <Mood>
      <Subject><segment start="18" end="29"/></Subject>
      <Finite><segment start="31" end="35"/></Finite>
    </Mood>
  </Clause>
</layer>
```

Figure 4: Ideational layer in stand-off XML format

AnnoLab offers a set of XQuery functions implementing the query operators we have defined for each of the tiers. That means, it offers all the power of XQuery and adds the capability of querying multiple annotation layers. Not only can it be used to query SFL-annotated resources, but also to analyze them simultaneously with non-SFL-annotations.

One of the points of interest in register analysis is thematic patterning. The Theme occupies the initial position in the clause and is thus a crucial indicator of what a text is about. Typically (i.e., most frequently), in English the Theme position is occupied by the Subject. If other choices are made (e.g., Adverbial), these appear marked. To investigate thematic patterns in an annotated corpus, we need to make reference to more than one layer of annotation, the layer encoding textual information and the layer encoding interpersonal information, in order to check how the Themes are filled. Figure 5 shows a sample cross-layer query that looks for Themes that are not coexistent with the Subject.

```
get-text(
  layer("Textual Metafunction")//Theme[
    not(overlaps(., layer("Interpersonal Metafunction")//Subject))])
```

Figure 5: Query example using segment overlaps

The first step of the query searches for all *Theme* elements in the *Textual Metafunction* layer. Each of these is tested for overlap with a *Subject* element from the *Interpersonal Metafunction* layer. The *overlaps* function collects all segments as-

sociated with the current *Theme* element and tests if one of them overlaps with any segment associated with a *Subject* element. All *Theme* elements that pass this test are dropped. The rest is finally passed to the location-based signal query operator *get-text*, which retrieves the text addressed by the segments associated with the remaining elements.

A number of pre-defined XQuery templates are included in AnnoLab, into which user queries (such as the one shown in Figure 5) are embedded and executed as subqueries. These templates take care of refining the search results and displaying them for example as a Keyword-in-Context concordance, cross-reference table or statistical report. Additionally, AnnoLab can display multiple annotation layers in a side-by-side view, aiding in the comparison of annotations as for instance required to analyze inter-annotator agreement.

AnnoLab is implemented in Java and is based on the open-source native XML database eXist [14] and the web development framework Cocoon [1].

5 Summary and Conclusions

In this paper we have discussed the issues involved in developing and deploying corpora based on Systemic Functional Linguistics (SFL). Our motivation for embarking on this endeavor is to have available a data set for carrying out quantitative linguistic analyses as required by register analysis, on the one hand, and to provide a basis for developing a model for probabilistic Systemic Functional Grammar and a corresponding parser, on the other.

The current state of our work can be summarized as follows.

Annotation. In contrast to comparable efforts in providing theory-rich corpus resources (e.g., the LinGo or Prague treebanks), we cannot draw on any automatic SFG-based parsing, so we have to carry out annotation manually. To create a high-quality resource, we go through the necessary steps for quality assurance (e.g., develop annotation guidelines, check interannotator agreement etc). Also, we are working towards an annotation tool that more adequately implements the particular needs of SFG analysis. As a supplement to manual annotation we employ PoS tagging and phrasal chunking. Also, we are exploring the possibilities of providing a more sophisticated structural analysis externally. Here, Dependency Grammar and Combinatory Categorial Grammar are suitable candidates, since they adopt structural segmentations compatible with those of SFG.

Representation. We have developed a data model for SFL-based corpora that conforms to the state-of-the-art in corpus representation. This model follows a multi-layer design that appropriately reflects the requirements of systemic-functional linguistic analysis and serves as a basis for corpus query [8].

Deployment in linguistic analysis. We have implemented a tool for linking multiple layers of annotation. This tool also provides a basic query functionality including query across layers [7]. Query is currently being developed further so as to cover additional analysis needs.

Acknowledgement. This work is supported by a research grant from DFG (*Deutsche Forschungsgemeinschaft*).

References

[1] Apache Cocoon Project. Available online from: http://cocoon.apache.org/, viewed June 2006.

[2] Olga Babko-Malaya, Ann Bies, Ann Taylor, Szuting Yi, Martha Palmer, Mitch Marcus, Seth Kulick, and Libin Shen. Issues in synchronizing the English Treebank and PropBank. In *Frontiers in Linguistically Annotated Corpora*, pages 70–77, Sydney, Australia, July 2006. COLING-ACL 2006, Association for Computational Linguistics.

[3] John A. Bateman. Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering*, 3(1):15–55, 1997.

[4] Steven Bird and Mark Liberman. A formal framework for linguistic annotation. *Speech Communication*, 33(1-2):23–60, 2001.

[5] Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Vidová-Hladká. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Kluwer Academic Publishers, Dordrecht, Boston, London, 2003.

[6] Thorsten Brants. TnT - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, pages 224–231, Seattle, WA, 2000.

[7] Richard Eckart. A framework for storing, managing and querying multi-layer annotated corpora. Diploma thesis, Technische Universität Darmstadt, Darmstadt, July 2006.

[8] Richard Eckart. Towards a modular data model for multi-layer annotated corpora. In *Proceedings of the COLING/ACL 2006 Main Conference Poster*

Sessions, pages 183–190, Sydney, Australia, July 2006. Association for Computational Linguistics.

[9] Steven Finch and Andrei Mikheev. A workbench for finding structure in texts. In W. Daelemans and M. Osborne, editors, *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, pages 372–379, Washington D.C., 1997.

[10] Anette Frank, Louisa Sadler, Josef van Genabith, and Andy Way. From treebank resources to LFG f-structures. Automatic f-structure annotation of treebank trees and CFGs extracted from treebanks. In Anne Abeillé, editor, *Treebanks. Building and using syntactically annotated corpora*, pages 367–389. Kluwer Academic Publishers, Dordrecht, Boston, London, 2003.

[11] Michael A. K. Halliday. *An introduction to functional grammar*. Arnold, London, 3. edition, 2004. revised by Matthiessen, Christian M. I. M.

[12] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[13] Christian M. I. M. Matthiessen. Frequency profiles of some basic grammatical systems: an interim report. In G. Thompson and S. Hunston, editors, *System and corpus: exploring connections*, pages 103–142. Equinox, London, 2006.

[14] Wolfgang Meier. eXist – Open Source native XML database. http://exist.sourceforge.net/index.html, 2006.

[15] Michael O'Donnell. From corpus to codings: Semi-automating the acquisition of linguistic features. In *Proceedings of the AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, Stanford University, California, March 1995.

[16] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. LinGO Redwoods. A rich and dynamic treebank for HPSG. In *First Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 139–149, Sozopol, Bulgaria, 2002.

[17] C. Michael Sperberg-McQueen and Claus Huitfeldt. Goddag: A data structure for overlapping hierarchies. In *DDEP/PODDP*, pages 139–160, Munich, 2000.

[18] Toru Sugimoto, Noriko Ito, Shino Iwashita, and Michio Sugeno. A computational framework for text processing based on systemic functional linguistics. In *1st Computational Systemic Functional Grammar Conference*, pages 2–11, University of Sydney, July 2005.

[19] Elke Teich, Silvia Hansen, and Peter Fankhauser. Representing and querying multi-layer corpora. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 228–237, Philadelphia, 11-13 December 2001. University of Pennsylvania.

[20] Canzhong Wu. *Modelling Linguistic Resources: a systemic-functional approach*. PhD thesis, Department of Linguistics, Macquarie University, Sydney, Australia, 2000.