Corpus annotation by generation

Elke Teich
TU Darmstadt
Darmstadt, Germany
teich@linglit.tu-darmstadt.de

John A. Bateman
Universität Bremen
Bremen, Germany
bateman@uni-bremen.de

Richard Eckart
TU Darmstadt
Darmstadt, Germany
eckart@linglit.tu-darmstadt.de

Abstract

As the interest in annotated corpora is spreading, there is increasing concern with using existing language technology for corpus processing. In this paper we explore the idea of using natural language *generation* systems for corpus annotation. Resources for generation systems often focus on areas of linguistic variability that are under-represented in analysis-directed approaches. Therefore, making use of generation resources promises some significant extensions in the kinds of annotation information that can be captured. We focus here on exploring the use of the KPML (Komet-Penman MultiLingual) generation system for corpus annotation. We describe the kinds of linguistic information covered in KPML and show the steps involved in creating a standard XML corpus representation from KPML's generation output.

1 Introduction

Many high-quality, theory-rich language processing systems can potentially be applied to corpus processing. However, the application of existing language technology, such as lexical and/or grammatical resources as well as parsers, turns out not to be as straightforward as one might think it should be. Using existing computational lexicons or thesauri, for instance, can be of limited value because they do not contain the domain-specific vocabulary that is needed for a particular corpus. Similarly, most existing grammatical resources for parsing have restricted *coverage* in precisely those areas of variation that are now most in need of corpus-supported investigation (e.g., predicate-argument structure, information structure, rhetorical structure). Apart from limited coverage, further issues that may impede the ready application of parsers in corpus processing include:

- *Annotation relevance*. Specialized, theory-specific parsers (also called 'deep parsers'; e.g., LFG or HPSG parsers) have been built with theoretical concerns in mind rather than appliability to unrestricted text. They may thus produce information that is not annotationally relevant (e.g., many logically equivalent readings of a single clause).

- *Usability*. Deep parsers are highly complex tools that require expert knowledge. The effort in acquiring this expert knowledge may be too high relative to the corpus processing task.

- *Completeness*. Simple parsers (commonly called 'shallow parsers'), on the other hand, produce only one type of annotationally relevant information (e.g., PoS, phrase/dependency structure). Other desirable kinds of information are thus lacking (e.g., syntactic functions, semantic roles, theme-rheme).

- *Output representation*. Typically, a parsing output is represented in a theory-specific way (e.g., in the case of LFG or HPSG parsers, a feature structure). Such output does not conform to the common practices in corpus representation.[1] Thus, it has to be mapped onto one of the standardly used data models for corpora (e.g., annotation graphs (Bird and Liberman, 2001) or multi-layer hierarchies (Sperberg-McQueen and Huitfeldt, 2001; Teich et al., 2001)) and transformed to a commonly employed format, typically XML.

[1] This is in contrast to the output representation of shallow parsers which have often been developed with the goal of corpus processing.

Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006, pages 86–93,
Sydney, July 2006.

In spite of these difficulties, there is a general consensus that the reward for exploring deep processing techniques to build up small to medium-scale corpus resources lies in going beyond the kinds of linguistic information typically covered by treebanks (cf. (Baldwin et al., 2004; Cahill et al., 2002; Frank et al., 2003)).

In this paper, we would like to contribute to this enterprise by adding a novel, yet complementary perspective on theory-rich, high-quality corpus annotation. In a reappraisal of the potential contribution of natural language generation technology for providing richly annotated corpora, we explore the idea of annotation by generation. Although this may at first glance seem counter-intuitive, in fact a generator, similar to a parser, creates rather complex linguistic descriptions (which are ultimately realized as strings). In our current investigations, we are exploring the use of these complex linguistic descriptions for creating annotations. We believe that this may offer a worthwhile alternative or extension of corpus annotation methods which may alleviate some of the problems encountered in parsing-based approaches.

The generation system we are using is the KPML (Komet-Penman MultiLingual; (Bateman, 1997)) system. One potential advantage of KPML over other generation systems and over many parsing systems is its multi-stratal design. The kinds of linguistic information included in KPML range from formal-syntactic (PoS, phrase structure) to functional-syntactic (syntactic functions), semantic (semantic roles/frames) and discoursal (e.g., theme-rheme, given-new). Also, since KPML has been applied to generate texts from a broad spectrum of domains, its lexicogrammatical resources cover a wide variety of registers—another potential advantage in the analysis of unrestricted text.

As well as our general concern with investigating the possible benefits of applying generation resources to the corpus annotation task, we are also more specifically concerned with a series of experiments involving the KPML system as such. Here, for example, we are working towards the construction of "treebanks" based on the theory of Systemic-Functional Linguistics (SFL; (Halliday, 2004)), so as to be able to empirically test some of SFL's hypotheses concerning patterns of instantiation of the linguistic system in authentic texts. Annotating the variety of linguistic categories given in SFL manually is very labor-intensive and an automated approach is clearly called for. We are also working towards a more detailed comparison of the coverage of the lexicogrammatical resources of KPML with those of parsing systems that are similarly theoretically-dedicated (e.g., the HPSG-based English Resource Grammar (ERG) (Copestake and Flickinger, 2002) contained in LinGO (Oepen et al., 2002)). Thus, the idea presented here is also motivated by the need to provide a basis for comparing grammar coverage across parsing and generation systems more generally.

The remainder of the paper is organized as follows. First, we present the main features of the KPML system (Section 2). Second, we describe the steps involved in annotation by generation, from the generation output (KPML internal generation record) to an XML representation and its refinement to an XML multi-layer representation (Section 3). Section 4 concludes the paper with a critical assessment of the proposed approach and a discussion of the prospects for application in the construction of corpora comparable in size and quality to existing treebanks (such as, for example, the Penn Treebank for English (Marcus et al., 1993) or the TIGER Treebank for German (Brants et al., 2002)). Since our description here has the status of a progress report of work still in its beginning stages, we cannot yet provide the results of detailed evaluation. In the final section, therefore, we emphasize the concrete steps that we are currently taking in order to be able carry out the detailed evaluations necessary.

2 Natural language generation with KPML

The KPML system is a mature grammar development environment for supporting large-scale grammar engineering work for natural language generation using multilingual systemic-functional grammars (Bateman et al., 2005). Grammars within this framework consist of large lattices of grammatical features, each of which brings constraints on syntactic structure. The features are also linked back to semantic configurations so that they can be selected appropriately when given a semantic specification as input. The result of generating with a systemic-functional grammar with KPML is then a rich feature-based representation distributed across a relatively simple structural backbone. Each node of the syntactic representation corresponds to an element of structure and

typically receives on the order of 50-100 linguistic features, called the *feature selection*. Since within systemic-functional grammars, it is the features of the feature selection that carry most of the descriptive load, we can see each feature selection as an exhaustive description of its associated syntactic constituent. Generation within KPML normally proceeds on the basis of a semantic input specification which triggers particular feature selections from the grammar via a mediating linguistic ontology.

The features captured in a systemic-functional generation resource are drawn from the four components of functional meaning postulated within systemic-functional grammar: the ideational, expressing content-related decisions, the logical, expressing logical dependencies, the interpersonal, expressing interactional, evaluative and speech act information, and the textual, expressing how each element contributes to an unfolding text. It is in this extremely rich combination of features that we see significant value in exploring the re-use of such grammars for annotation purposes and corpus enrichment.

For annotation purposes, we employ some of the alternative modes of generation that are provided by the full grammar development environment—it is precisely these that allow for ready incorporation and application within the corpus annotation task. One of the simplest ways in which generation can be achieved during grammar development, for example, is by directly selecting linguistic features from the grammar. This can therefore mimic directly the task of annotation: if we consider a target sentence (or other linguistic unit) to be annotated, then selecting the necessary features to generate that unit is equivalent to annotating that unit in a corpus with respect to a very extensive set of corpus annotation features.

Several additional benefits immediately acrue from the use of a generator for this task. First, the generator *actually constructs the sentence* (or other unit) as determined by the feature selection. This means that it is possible to obtain immediate feedback concerning the correctness and completeness of the annotation choices with respect to the target. A non-matching structure can be generated if: (a) an inappropriate linguistic feature has been selected, (b) the linguistic resources do not cover the target to be annotated, or (c) a combination of these. In order to minimise the influence of (b), we only work with large-scale grammatical resources whose coverage is potentially sufficient to cover most of the target corpus. Further corpus instances that lie beyond the capabilities of the generation grammar used are an obvious source of requirements for extensions to that grammar.

Second, the architecture of the KPML system also allows for other kinds of annotation support. During grammar development it is often required that guidance is given directly to the semantics-grammar linking mappings: this is achieved by providing particular 'answers' to pre-defined 'inquiries'. This allows for a significantly more abstract and 'intention'-near interaction with the grammatical resource that can be more readily comprehensible to a user than the details of the grammatical features. This option is therefore also available for annotation.

Moreover, the semantic specifications used rely on a specified linguistic ontology that defines particular semantic types. These types can also be used directly in order to constrain whole collections of grammatical features. Providing this kind of guidance during annotation can also, on the one hand, simplify the process of annotation while, on the other, produce a semantic level of annotation for the corpus.

In the following sections, we see a selection of these layers of information working in annotation in more detail, showing that the kinds of information produced during generation corresponds extremely closely to the kinds of rich annotations currently being targetted for sophisticated corpus presentation.

3 Creating corpus annotations from KPML output

3.1 KPML output

The output produced by KPML when being used for generation is a recursive structure with the chosen lexical items at the leaves. Figure 1 shows the output tree for the sample sentence "However they will step up their presence in the next year".

The nodes of this structure may be freely annotated by the user or application system to contain further information: e.g., for passing through hyperlinks and URLs directly with the semantics when generating hypertext. Most users simply see the result of flattening this structure into a string: the generated sentence or utterance.

This result retains only a fraction of the in-

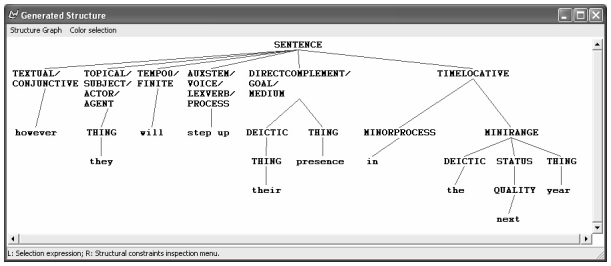

Figure 1: Tree generated by KPML

formation that is employed by the generator during generation. Therefore, since we are using the grammar development environment rather than simply the generator component, we also have the possibility of working directly with the internal structures that KPML employs for display and debugging of resources during development. These internal structures contain a complete record of the information provided to the generation process and the generator decisions (including which grammatical features have been selected) that have been made during the construction of each unit. This internal record structure is again a recursive structure corresponding directly to the syntactic structure of the generated result and with each node having the information slots:

```
constituent:
{identifier, \\ unique id for the unit
 concept,    \\ link to the semantic concept expressed
 spelling,   \\ the substring for this portion of structure
 gloss,      \\ a label for use in inter-lineal glosses
 features,   \\ the set of grammatical features for this unit
 lexeme,     \\ the lexeme chosen to cover this unit (if any)
 annotation, \\ user-specified information
 functions   \\ the grammatical functions the unit expresses
}
```

An extract from such an internal record structure encoded in XML is given in the Appendix (5.1).

To support annotation, we make use of the XML-export capabilities of KPML (cf. (Bateman and Hartley, 2000)) in order to provide these completed structures in a form suitable for passing on to the next stage of corpus annotation within an XML-based multi-layer framework.

3.2 XML multi-layer representation

Systemic-functional analysis is inherently multi-dimensional in that SFL adopts more than one view on a linguistic unit. Here, we focus on three annotationally relevant dimensions: axis (features and functions), unit (clause, group/phrase, word, morpheme) and metafunction (ideational, logical, interpersonal and textual). Each metafunction may chunk up a given string (e.g., a clause unit) in

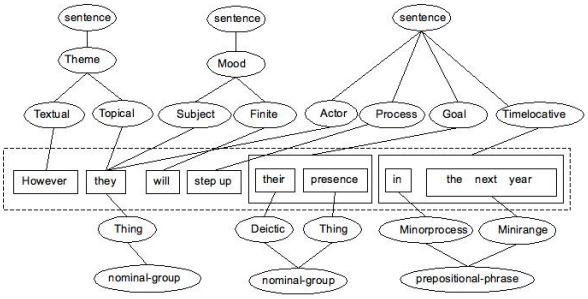

Figure 2: Generation output viewed as multi-layer annotation

```
<sfglayer metafunction="IDEATIONAL">
  However,
  <segment functions="AGENT">they</segment>
  will step up
  <segment functions="DIRECTCOMPLEMENT GOAL MEDIUM">
    their presence
  </segment>
  <segment functions="TIMELOCATIVE">
    in the next year
  </segment>
  .
</sfglayer>
```

Figure 3: Metafunction+Function layers

different ways, thus potentially creating overlapping hierarchies. This is depicted schematically for the running example in Figure 2. For instance, in this example, according to the textual metafunction, “however they” constitutes a segment (Theme) and according to the interpersonal metafunction, “they will” constitutes another segment (Mood).

In order to be able to use the KPML output for annotation purposes, we adopt a multi-layer model that allows the representation of these different descriptional dimensions as separate layers superimposed on a given string (cf. (Teich et al., 2005)). The transformation from the KPML output to the concrete multi-layer model adopted is defined in XSLT.

From the KPML internal record structure we use the information slots of identifier, spelling, features, and functions. Each entry in the function slot is associated with one metafunctional aspect. For each metafunctional aspect, an annotation layer is created for each constituent unit (e.g., a clause) holding all associated functions together with the substrings they describe (see Figure 3 for the ideational functions contained in the clause in the running example).

An additional layer holds the complete constituent structure of the clause (cf. Figure 4 for the corresponding extract from the running example),

```
<constituent unit="-TOP-"
  selexp="LEXICAL-VERB-TERM-RESOLUTION...">
  <token features="HOWEVER">However,</token>
  <constituent unit="TOPICAL"
    selexp="THEY-PRONOUN...">
    <token features="THEY PLURAL-FORM">they</token>
  </constituent>
  <token features="OUTCLASSIFY-REDUCED...">will</token>
  <token features="DO-VERB...">step up</token>
  <constituent  unit="DIRECTCOMPLEMENT"
    selexp="NOMINAL-TERM-RESOLUTION OBLIQUE...">
    <constituent unit="DEICTIC"
      selexp="THEIR GENITIVE NONSUPERLATIVE...">
      <token features="THEIR PLURAL-FORM">their</token>
    </constituent>
    <token features="...COMMON-NOUN...">presence</token>
  </constituent>
  <constituent unit="TIMELOCATIVE"
    selexp="IN STRONG-INCLUSIVE UNORDERED...">
    <token features="IN">in</token>
    <constituent unit="MINIRANGE"
      selexp="NOMINAL-TERM-RESOLUTION...">
      <token features="THE">the</token>
      <constituent unit="STATUS"
        selexp="QUALITY-TERM-RESOLUTION...">
        <token features="...ADJECTIVE">next</token>
      </constituent>
      <token features="...COMMON-NOUN...">year .</token>
    </constituent>
  </constituent>
</constituent>
```

Figure 4: Constituent+Feature layer

i.e., the phrasal constituents and their features:

```
<constituent unit="..." selexp="...">
</constituent>
```

and the tokens and their (lexical) features:

```
<token features="..."> ... </token>
```

Thus, the KPML generation output, which directly reflects the trace of the generation process, is reorganized into a meaningful corpus representation. Information not relevant to annotation can be ignored without loss of information concerning the linguistic description. The resulting representation for the running example is shown in the Appendix (5.2).[2]

4 Discussion

Although it is clear that the kind of informational structures produced during generation with more developed KPML grammars align quite closely with that targetted by sophisticated corpus annotation, there are several issues that need to be addressed in order to turn this process into a practical annotation alternative. Those which we are currently investigating centre around usability and coverage.

[2]To improve readability, we provide the integrated representation rather than the stand-off representation which aligns the different layers by using character offsets.

Usability/effort. Users need to be trained in providing information to guide the generation process. This guidance is either in the form of direct selections of grammatical features, in which case the user needs to know when the features apply, or in the form of semantic specifications, in which case the user needs information concerning the appropriate semantic classification according to the constructs of the linguistic ontology. One of the methods by which the problem of knowing the import of grammatical features may be alleviated is to link each feature with sets of already annotated/generated corpus examples. Thus, if a user is unsure concerning a feature, she can call for examples to be displayed in which the particular linguistic unit carrying the feature is highlighted. Even more useful is a further option which shows not only examples containing the feature, but *contrasting* examples showing where the feature has applied and where it has not. This provides users with online training during the use of the system for annotation. The mechanisms for showing examples and contrasting sets of generated sentences for each feature were originally provided as part of a teaching aid built on top of KPML: this allows students to explore a grammar by means of the effects that each set of contrasting features brings for generated structures. For complex grammars this appears to offer a viable alternative to precise documentation—especially for less skilled users.

Coverage. When features have been selected, it may still be the case that the correct target string has not been generated due to limited coverage of grammar and/or semantics. This is indicative of the need to extend the grammatical resources further. A further alternative that we are exploring is to allow users to specify the correspondence between the units generated and the actual target string more flexibly. This is covered by two cases: (i) that additional material is in the target string that was not generated, and (ii) that the surface order of constituents is not exactly that produced by the generator. In both cases we can refine the stand-off annotation so that the structural result of generation can be linked to the actual string. Thus manual correction consists of minor alignment statements between generated structure and string.

Certain other information that may not be available to the generator, such as lexical entries, can be constructed semi-automatically on-the-fly, again

using the information produced in the generation process (i.e., by collecting the lexical classification features and adding lexemes containing those features). This method can be applied for all open word classes.

Next steps. In our future work, we will be carrying out an extensive annotation experiment with the prediction that annotation time is not higher than for interactive annotation from a parsing perspective. TIGER, for example, reports 10 minutes per sentence as an average annotation time. We expect an experienced KPML user to be significantly faster because the process of generation or feature selection explicitly leads the annotator through precisely those features that are relevant and possible given the connectivity of the feature lattice defined by the grammar. Annotation then proceeds first by selecting the features that apply and then by aligning the generated structure with the corpus instance: both potentially rather rapid stages. Also, we would expect to achieve similar coverage as reported by (Baldwin et al., 2004) for ERG when applied to a random 20,000 string sample of the BNC due to the coverage of the existing grammars.

The results of such investigations will be SFL-treebanks, analogous to such treebanks produced using dependency approaches, LFG, HPSG, etc. These treebanks will then support the subsequent learning of annotations for automatic processing.

Acknowledgment. This work was partially supported by *Hessischer Innovationsfond* of TU Darmstadt and PACE (Partners for the Advancement of Collaborative Engineering Education: `www.pacepartners.org`).

References

T. Baldwin, E. M. Bender, D. Flickinger, A. Kim, and S. Oepen. 2004. Road-testing the EnglishResource Grammar over the British National Corpus. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC) 2004*, Lisbon, Portugal.

J. A. Bateman and A. F. Hartley. 2000. Target suites for evaluating the coverage of text generators. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, Athens, Greece.

J. A. Bateman, I. Kruijff-Korbayová, and G.-J. Kruijff. 2005. Multilingual resource sharing across both related and unrelated languages: An implemented, open-source framework for practical natural language generation. *Research on Language and Computation*, 3(2):191–219.

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

S. Bird and M. Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(1-2):23–60.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

A. Cahill, M. McCarthy, J. van Genabith, and A. Way. 2002. Automatic annotation of the Penn-Treebank with LFG f-structure information. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC) 2002*, Las Palmas, Spain.

A. Copestake and D. Flickinger. 2002. An open-source grammar development environment and broad coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*, Athens, Greece.

A. Frank, L. Sadler, J. van Genabith, and A. Way. 2003. From treebank resources to LFG f-structures. Automatic f-structure annotation of treebank trees and CFGs extracted from treebanks. In A. Abeille, editor, *Treebanks. Building and using syntactically annotated corpora*, pages 367–389. Kluwer Academic Publishers, Dordrecht, Boston, London.

MAK Halliday. 2004. *Introduction to Functional Grammar*. Arnold, London.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

S. Oepen, E. Callahan, D. Flickinger, C. D. Manning, and K. Toutanova. 2002. LinGO Redwoods. A rich and dynamic treebank for HPSG. In *Workshop on Parser Evaluation, 3rd International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Spain.

C.M. Sperberg-McQueen and C. Huitfeldt. 2001. GODDAG: A Data Structure for Overlapping Hierarchies. In *Proceedings of PODDP'00 and DDEP'00*, New York.

E. Teich, S. Hansen, and P. Fankhauser. 2001. Representing and querying multi-layer corpora. In *Proceedings of the IRCS Workshop on Linguistic Databases*, University of Pennsylvania, Philadelphia.

E. Teich, P. Fankhauser, R. Eckart, S. Bartsch, and M. Holtz. 2005. Representing SFL-annotated corpus resources. In *Proceedings of the 1st Computational Systemic Functional Workshop*, Sydney, Australia.

5 Appendix

5.1 Extract from generation record (clause level)

```
<example>
 <name>REUTERS29</name>
  <generatedForm>However, they will step up their presence in the next year.</generatedForm>
  <targetForm>But they will step up their presence in the next year.</targetForm>
  <structures><constituent id="G3324" semantics="STEP-3278">
   <functions>
     <function metafunction="UNKNOWN">SENTENCE</function></functions>
   <features/>
  <subconstituents><constituent id="G3308" semantics="RR62-3289">
   <functions>
     <function metafunction="TEXTUAL">TEXTUAL</function>
     <function metafunction="TEXTUAL">CONJUNCTIVE</function></functions>
   <features>
      <f>HOWEVER</f></features>
  <subconstituents><string>However,</string></subconstituents>
  </constituent><constituent id="G3310" semantics="PERSON-3291">
   <functions>
     <function metafunction="TEXTUAL">TOPICAL</function>
     <function metafunction="INTERPERSONAL">SUBJECT</function>
     <function metafunction="UNIFYING">ACTOR</function>
     <function metafunction="IDEATIONAL">AGENT</function></functions>
   <features/>
  <subconstituents><constituent id="G3309" semantics="PERSON-3291">
   <functions>
     <function metafunction="LOGICAL">THING</function></functions>
   <features>
      <f>THEY</f>
      <f>PLURAL-FORM</f></features>
  <subconstituents><string>they </string></subconstituents>
  </constituent>
  </subconstituents>
  </constituent><constituent id="G3311" semantics="ST59-3280-3297-3302">
   <functions>
     <function metafunction="LOGICAL">TEMPO0</function>
     <function metafunction="INTERPERSONAL">FINITE</function></functions>
   <features>
      <f>OUTCLASSIFY-REDUCED</f>
      <f>OUTCLASSIFY-NEGATIVE-AUX</f>
      <f>FUTURE-AUX</f>
      <f>PLURAL-FORM</f>
      <f>THIRDPERSON-FORM</f></features>
  <subconstituents><string>will </string></subconstituents>
  </constituent><constituent id="G3312" semantics="STEP-3278">
   <functions>
     <function metafunction="UNIFYING">AUXSTEM</function>
     <function metafunction="LOGICAL">VOICE</function>
     <function metafunction="LOGICAL">LEXVERB</function>
     <function metafunction="LOGICAL">PROCESS</function></functions>
   <features>
      <f>DO-VERB</f>
      <f>EFFECTIVE-VERB</f>
      <f>DISPOSAL-VERB</f>
      <f>STEM</f></features>
  <subconstituents><string>step up </string></subconstituents>
  </constituent><constituent id="G3316" semantics="PRESENCE-3292-3306">
   <functions>
     <function metafunction="IDEATIONAL">DIRECTCOMPLEMENT</function>
     <function metafunction="IDEATIONAL">GOAL</function>
     <function metafunction="IDEATIONAL">MEDIUM</function></functions>
  </constituent></subconstituents></constituent></structures>
  <selectionexpressions>
     <selexp sem="STEP-3278"><unit>-TOP-</unit><f>LEXICAL-VERB-TERM-RESOLUTION</f>
      <f>DO-NEEDING-VERBS</f><f>AUXSTEM-VOICE</f><f>REAL</f><f>NON-MOTION-CLAUSE</f>
      <f>PLURAL-FINITE</f><f>PLURAL-SUBJECT</f><f>TOPICAL-INSERT</f> ...
     </selexp>
     <selexp>...</selexp>
     ...
  </selectionexpressions>
</example>
```

5.2 Multi-layer representation of generation record

Metafunction+Function layers

```
<sfglayer metafunction="UNKNOWN">
  <segment functions="SENTENCE">
    However, they will step up their presence in the next year .
  </segment>
</sfglayer>

<sfglayer metafunction="UNIFYING">
  However,
  <segment functions="ACTOR">they</segment>
  will
  <segment functions="AUXSTEM">step up</segment>
  their presence in the next year .
</sfglayer>

<sfglayer metafunction="TEXTUAL">
  <segment functions="TEXTUAL CONJUNCTIVE">However,</segment>
  <segment functions="TOPICAL">they</segment>
  will step up their presence in the next year .
</sfglayer>
```

```
<sfglayer metafunction="LOGICAL">
  However,
  <segment functions="THING">they</segment>
  <segment functions="TEMPO0">will</segment>
  <segment functions="VOICE LEXVERB PROCESS">step up</segment>
  <segment functions="THING">their</segment>
  <segment functions="THING">presence</segment>
  in the
  <segment functions="QUALITY">next</segment>
  <segment functions="THING">year .</segment>
</sfglayer>

<sfglayer metafunction="INTERPERSONAL">
  However,
  <segment functions="SUBJECT">they</segment>
  <segment functions="FINITE">will</segment>
  step up
  <segment functions="DEICTIC">their</segment>
  presence in
  <segment functions="DEICTIC">the</segment>
  next year .
</sfglayer>

<sfglayer metafunction="IDEATIONAL">
  However,
  <segment functions="AGENT">they</segment>
  will step up
  <segment functions="DIRECTCOMPLEMENT GOAL MEDIUM">
    their presence
  </segment>
  <segment functions="TIMELOCATIVE">
    <segment functions="MINORPROCESS">in</segment>
    <segment functions="MINIRANGE">
      the
      <segment functions="STATUS">next</segment>
      year .
    </segment>
  </segment>
</sfglayer>
```

Constituent+Feature layer

```
<constituent id="G3324" unit="-TOP-"
  selexp="LEXICAL-VERB-TERM-RESOLUTION DO-NEEDING-VERBS AUXSTEM-VOICE REAL NON-MOTION-CLAUSE TOPICAL-INSERT ...">
  <token features="HOWEVER">However,</token>
  <constituent id="G3310" unit="TOPICAL"
    selexp="THEY-PRONOUN NONDEMONSTRATIVE-SPECIFIC-PRONOUN NOMINATIVE NONSUPERLATIVE NONREPRESENTATION NONPARTITIVE ...">
    <constituent id="G3309" unit="TOPICAL">
      <token features="THEY PLURAL-FORM">they</token>
    </constituent>
  </constituent>
  <token
    features="OUTCLASSIFY-REDUCED OUTCLASSIFY-NEGATIVE-AUX FUTURE-AUX PLURAL-FORM THIRDPERSON-FORM">
    will
  </token>
  <constituent id="G3312" unit="-TOP-">
    <token features="DO-VERB EFFECTIVE-VERB DISPOSAL-VERB STEM">
      step up
    </token>
  </constituent>
  <constituent id="G3316" unit="DIRECTCOMPLEMENT"
    selexp="NOMINAL-TERM-RESOLUTION OBLIQUE NONSUPERLATIVE NONREPRESENTATION NONPARTITIVE NONQUANTIFIED NOMINAL-GROUP ...">
    <constituent id="G3314" unit="DEICTIC"
      selexp="THEIR GENITIVE NONSUPERLATIVE NONREPRESENTATION NONPARTITIVE NONQUANTIFIED  NOMINAL-GROUP ...">
      <constituent id="G3313" unit="DEICTIC">
        <token features="THEIR PLURAL-FORM">their</token>
      </constituent>
    </constituent>
    <constituent id="G3315" unit="DIRECTCOMPLEMENT">
      <token
        features="OUTCLASSIFY-PROPERNOUN NOUN COMMON-NOUN COUNTABLE SINGULAR-FORM NOUN">
        presence
      </token>
    </constituent>
  </constituent>
  <constituent id="G3323" unit="TIMELOCATIVE"
    selexp="IN STRONG-INCLUSIVE UNORDERED TEMPORAL-PROCESS LOCATION-PROCESS SPATIO-TEMPORAL-PROCESS PREPOSITIONAL-PHRASE ...">
    <token features="IN">in</token>
    <constituent id="G3322" unit="MINIRANGE"
      selexp="NOMINAL-TERM-RESOLUTION OBLIQUE NONSUPERLATIVE NONREPRESENTATION NONPARTITIVE NONQUANTIFIED NOMINAL-GROUP ...">
      <token features="THE">the</token>
      <constituent id="G3320" unit="STATUS"
        selexp="QUALITY-TERM-RESOLUTION SIMPLEX-QUALITY NOTINTENSIFIED NONSCALABLE CONGRUENT-ADJECTIVAL-GROUP ...">
        <constituent id="G3319" unit="STATUS">
          <token features="OUTCLASSIFY-DEGREE-ADJ ADJ-NEUTRAL-FORM ADJECTIVE">
            next
          </token>
        </constituent>
      </constituent>
      <constituent id="G3321" unit="MINIRANGE">
        <token features="OUTCLASSIFY-PROPERNOUN NOUN COMMON-NOUN COUNTABLE SINGULAR-FORM NOUN">
          year .
        </token>
      </constituent>
    </constituent>
  </constituent>
</constituent>
```