Amalgam:

A machine-learned generation module

Michael Gamon, Eric Ringger, and Simon Corston-Oliver
{mgamon, ringger, simonco}@microsoft.com

11 June 2002

Technical Report

MSR-TR-2002-57

Microsoft Research
One Microsoft Way
Redmond WA 98052
USA

Abstract

Amalgam is a novel system for sentence realization during natural language generation. Amalgam takes as input a logical form graph, which it transforms through a series of modules involving machine-learned and knowledge-engineered sub-modules into a syntactic representation from which an output sentence is read. Amalgam constrains the search for a fluent sentence realization by following a linguistically informed approach that includes such component steps as raising, labeling of phrasal projections, extraposition of relative clauses, and ordering of elements within a constituent.

In this technical report we describe the architecture of Amalgam based on a complete implementation that generates German sentences. We describe several linguistic phenomena, such as relative clause extraposition, that must be handled in order to successfully generate German.

1 Introduction

We describe the architecture of a novel sentence realization component, Amalgam ("A Machine-Learned Generation Module"). Amalgam is a module within the German NLPWIN system at Microsoft Research.

The need for a sentence realization module arose in the context of on-going research into machine translation (Richardson et al. 2001a, Richardson et al. 2001b). Sentences in a source language are analyzed to logical forms. These logical forms are transferred to the target language, and must then be realized as fluent sentences.

For some target languages we already had mature, high quality knowledge-engineered sentence realization modules (Aikawa et al. 2001a, 2001b). For German, we did not already have a sentence realization module. We therefore embarked on the undertaking described in this technical report, namely to produce an empirically-based sentence realization module by employing machine learning techniques as much as possible.

As a first step towards a generation module that is usable in machine translation contexts, we created a module that generates German output strings from German input strings by roundtrip of analysis and subsequent generation as a proof-of-concept. The main focus of this report is on the German-to-German generation approach, although we briefly discuss first results of the application of the Amalgam system in machine translation in section 10. The advantage of approaching the task from the German-to-German generation perspective is that evaluation of the system is straightforward. The input string goes through syntactic and semantic analysis, a logical form representation is produced, and an output string is generated from that logical form representation by Amalgam. If the output string is identical to the input string, Amalgam has performed flawlessly. The farther the output string is from the input string, the worse Amalgam has performed. This is, of course, an over-simplified view, given that there is often more than one good German sentence that would represent a logical form faithfully and fluently (see, for example, the discussion of relatively free constituent order in German in section 3.4).

Despite this caveat, the German-to-German approach has provided a good starting point for the development of the first prototype of Amalgam.

Amalgam has been described in published papers (Corston-Oliver et al. 2002, Gamon et al. 2002a, Gamon et al. 2002b). Our goal in this technical report is to provide a complete description of the architecture and implementation of Amalgam, beyond the level of detail that is customary in conference proceedings or journal papers. We hope that this technical report will provide answers to some of the questions that inevitably arise when reading published descriptions of natural language processing systems, including the following questions: Exactly which features are used? How are the features extracted? How much work is performed by the knowledge-engineered module mentioned in passing?

2 Prior work in sentence realization

Reiter (1994) surveys the major natural language generation systems of the late 1980s through the mid-1990s: FUF (Elhadad 1992), IDAS (Reiter at al. 1992), JOYCE (Rambow and Korelsky 1992), PENMAN (Penman 1989) and SPOKESMAN (Meteer 1989). Each of these systems has a different theoretical underpinning: unification grammar in the case of FUF (Kay 1979), a generalized reasoning system (Reiter and Mellish 1992) in the case of IDAS, Meaning-Text theory (Mel'čuk 1988) in the case of JOYCE, Hallidayan Systemic Functional Linguistics (Halliday 1985) and Rhetorical Structure Theory (Mann and Thompson 1988) in the case of PENMAN, Tree-Adjoining Grammar (Joshi 1987) in the case of SPOKESMAN. Despite their diverse theoretical underpinnings, Reiter draws attention to the fact that a consensus appeared to have emerged concerning the appropriate architecture for a natural language generation system. All systems had a module that performed content determination, mapping input specifications of content onto a semantic form, followed by a module that performed sentence planning. Output was performed by a surface generation module (what we refer to below as a sentence realization module) that made use of a morphology module and a module that performed formatting of the output text. All the systems that Reiter surveyed generate English text only.

Reiter draws speculative analogies between the consensus architecture and the evidence for modularity of language in the human brain based on language impairment of individuals with various types of brain injury, and suggests that the engineering trade-offs made during system implementation might mirror evolutionary forces at work in the development of human language.

The dominant paradigm for natural language generation systems up until the mid-1990s was that of knowledge engineering. Computational linguists would explicitly code strategies for stages ranging from planning texts and aggregating content into single sentences to choosing appropriate forms of referring expressions performing morphological inflection and formatting output. This research path has yielded several mature broad-coverage systems and is still being actively pursued today; see, for example, the sentence realization modules described in Aikawa et al. (2001).

Since the mid 1990s there has been increasing interest in the application of statistical and machine learning techniques to the various stages of natural language generation. This research has ranged from learning plans for high-level planning of texts and dialogues (Zukerman et al. 1998, Duboue and McKeown 2001) or ensuring that the macro properties of generated texts such as the distribution of sentence lengths and lexical variety mirror the properties of naturally occurring texts (Oberlander and Brew, 2000) to sentence planning (Walker et al., 2001), lexical selection (Bangalore and Rambow 2000b), selection of the appropriate form for a referring expression (Poesio et al 1999), determining grammatical relations (Corston-Oliver 2000) and selecting the appropriate word order (Langkilde and Knight 1998a, Langkilde and Knight 1998b, Bangalore and Rambow 2000a).

It is often the case in the natural language generation literature that descriptions of the higher level aspects of natural language generation gloss over issues associated with sentence realization. Walker et al. (2001), for example, characterize realization in this way:

> "During realization, the abstract linguistic resources chosen during sentence planning are transformed into a surface linguistic utterance by adding function words (such as auxiliaries and determiners), inflecting words, and determining word order. This phase is not a planning phase in that it only executes decisions made previously." (Walker et al. 2001)

In typical implementations, however, once the planning stages, *sensu stricto*, have finished there remain myriad encoding decisions to be made and selections among alternative formulations to be performed. Increasingly, machine-learned techniques are being brought to bear on these tasks.

Two recent systems, the Nitrogen system (Langkilde and Knight 1998a, Langkilde and Knight 1998b) and the FERGUS system (Bangalore and Rambow 2000a) are sufficiently similar to Amalgam to warrant extended discussion.

The Nitrogen system (Langkilde and Knight 1998a, Langkilde and Knight 1998b) uses word bigrams instead of deep symbolic knowledge to decide among alternative output sentences. The input to Nitrogen can range from rather abstract semantic representations to more fully-specified syntactic representations. Inputs are given in the Abstract Meaning Representation, based on the Penman Sentence Plan Language (Penman 1989). Two sets of knowledge-engineered rules operate on the input specification to produce candidate output sentences. One set of rules performs one-to-many mappings from underspecified semantic representations to possible syntactic formulations, fleshing out information such as definiteness and number that might be missing in practical generation contexts such as Japanese to English machine translation (Knight et al 1995). The second set of rules, which includes sensitivity to the target domain, transforms the representations produced by the first module to yield still more candidate sentences. The candidate sentences are compactly represented as a word lattice. Word bigrams are used to score and find the optimal traversal of the lattice, yielding the best-ranked output

sentence. Morphological inflection is performed by simple table lookup, apparently during the production of candidate sentences.

Langkilde and Knight (1998a) present worked examples that illustrate the importance of the bigram filtering. One input semantic form includes five lexical nodes in such relationships as AGENT, DESTINATION, and PATIENT. The word lattice that results contains more than eleven million possible paths, with the top-ranked candidate "Visitors who came in Japan admire Mount Fuji." Another worked example, for which the semantic representation is not given, appears to involve two content words that are transformed into a word lattice containing more than 155,000 paths to yield the top-ranked candidate "I cannot betray their trust."

Clearly, there is an important interaction between the knowledge-engineered components that propose candidates and the bigram filtering. If the knowledge-engineered components are too conservative, an optimal rendering will not be proposed, forcing the bigram filtering component to choose among sub-optimal candidates. On the other hand, if the knowledge-engineered component proposes too many candidates, the search through the lattice may become so time-consuming as to be impractical. Unfortunately, Langkilde and Knight do not give more details about the knowledge-engineered components. One wonders how many rules there are, how many rules must be added for a new domain, and what level of expertise is required to write a rule.

The use of bigrams is problematic, as Langkilde and Knight acknowledge. Bigrams are unable to capture dependencies among non-contiguous words, a fact that is perhaps mitigated by the observation that in practice, for English at least, syntactic dependencies most often obtain between adjacent elements (Stolcke 1997). Increasing the number of terms to trigrams or higher-order n-grams raises the familiar specter of paucity of data. Furthermore, as Langkilde and Knight observe, many linguistic relationships are binary in nature, and therefore not efficiently represented using trigrams. To overcome some of the deficiencies of the bigram language model applied to a word lattice, Langkilde (nd) proposes using a parse forest to represent the output candidates. The evaluation metric that she intends to develop would combine information from the syntactic representations and the language model.

It is unclear how feasible it would be to generate candidate sentences and then filter them when generating German. For English, Langkilde and Knight use table lookup to add morphological variants of content words to the mix. Since English inflectional morphology is relatively simple, this does not adversely explode the search space. When we consider the richer inflectional morphology of German, however, this simple table lookup does not appear practical. Whereas English has a single definite article, *the*, German has six inflected forms (*der*, *die*, *das*, etc)[1]. Similarly, English adjectives can be inflected only for degree (e.g., *big*, *bigger*, *biggest*), whereas German adjectives additionally distinguish three lexical genders, four cases, and singular vs. plural. If the search space becomes large using table-based lookup to propose additional nodes in a

[1] For a subset of all possible inflected German determiners, see the variants listed in Figure 34.

word lattice for English, it would become intractable for a language such as German with richer inflectional morphology.

Recent research has demonstrated the usefulness of syntactic information in overcoming the inadequacies of bigrams. Ratnaparkhi (2000) demonstrates dramatic improvements in selecting appropriate output templates for the air travel domain when conditioning on syntactic dependencies versus conditioning on trigrams.

Further validation of the usefulness of syntax during sentence realization can be seen in the FERGUS system (Bangalore and Rambow 2000a). Bangalore and Rambow augment the work of Langkilde and Knight by adding a tree-based stochastic model and a traditional tree-based syntactic grammar. Bangalore and Rambow take as input a dependency tree. A stochastic tree model chooses TAG trees for the nodes in the dependency tree. The resulting TAG analysis is then "unraveled" to produce a lattice of compatible linearizations. Selection among competing linearizations is performed by a "Linear Precedence Chooser" which selects the most likely linearization given a suitable language model.

To date there have been no published descriptions of the application of machine learning to the problems of morphological realization or formatting for natural language generation, although presumably inflectional morphology that had been learnt automatically (e.g., Goldsmith 2001) could subsequently be applied during generation.

3 Properties of German

The German language exhibits a number of properties that are very different from English, despite the fact that the two languages are relatively closely related. These properties pose challenges to a sentence realization component which go beyond what an English sentence realizer would have to account for. For us, this poses the interesting task of making the overall design of Amalgam flexible enough to deal with these phenomena, and as a result, flexible enough to be applicable to more languages. It also protects us from the myopia of NLP solutions predicated on properties of English, such as the paucity of inflectional morphology and the relative rigidity of constituent order.

In this section, we present a brief overview of the important characteristics of German. It should be understood that this is by no means a complete list of the properties in which German differs from English. We focus on a handful of properties crucial in sentence realization. We contrast these properties with English, to emphasize the typological differences between the two languages, particularly for the benefit of English speakers who are not familiar with German.

3.1 The Position of the Verb in German

One of the most striking properties of German, painfully familiar to anyone who has learned German as a foreign language, is the distribution of verbs in main and subordinate clauses. In fact, most descriptive accounts of German syntax are based on a topology of the German sentence that treats the position of the verb as the fixed frame around which other syntactic constituents are organized in a relatively free order (cf.

Eisenberg 1999, Engel 1996). The general frame of the German sentence is shown in Figure 1.

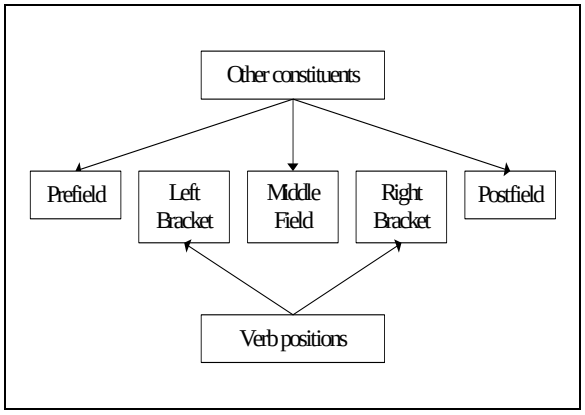


Figure 1: The topological model of the German sentence

The important facts to note about this topological model are:

- The Prefield contains at most one constituent
- The Left Bracket contains:
  - the finite verb OR
  - a subordinating conjunction OR
  - a relative pronoun/relative expression
- The Middle Field contains any number of constituents
- The Right Bracket contains all the verbal material that is not present in the Left Bracket. If the finite verb is in the Left Bracket, then the Right Bracket contains the non-finite verbs. If the Left Bracket is occupied by a subordinating conjunction or relative expression, the Right Bracket contains all the verbs.
- The Postfield contains:
  - clausal complements
  - subordinate clauses
  - extraposed material (e.g., relative clauses extraposed from the Middle Field)
  - other constituents

The position of the verb in German is rigidly fixed. Errors in the positioning of the verb will result in gibberish, while most permutations within Prefield, Middle Field and Postfield will at worst result in less fluent output. Depending on the position of the finite

verb, German sentences and verb phrases are often classified as being "verb-initial", "verb-second" or "verb-final". In verb-initial clauses, the finite verb is in initial position (e.g., in the imperative example in Figure 2). Verb-second sentences contain material in the Prefield, and a finite verb in the Left Bracket. Verb-final sentences (such as the complement clause in Figure 2) contain no verbal element in the Left Bracket (usually because the Left Bracket is occupied by a subordinating conjunction or a relative pronoun). Figure 2 illustrates the above generalizations with some examples. German text is in italics, glosses are given below each word, and free translations are given in quotes.

Prefield	**Left Bracket**	**Middle Field**	**Right Bracket**	**Postfield**
Main clauses (declarative)				
Hans Hans	*sieht* sees	*das Auto* the car		
"Hans sees the car"				
Hans Hans	*hat* has	*das Auto* the car	*gesehen* seen	
"Hans has seen the car"				
Hans Hans	*gibt* gives	*das Buch* the book	*ab* PREFIX	
"Hans returns the book"				
Hans Hans	*wird* will	*das Auto* the car	*gesehen haben* seen have	
"Hans will have seen the car"				
Hans Hans	*hat* has	*das Auto* the car	*gesehen* seen	*das er kaufen möchte* that he buy wants
"Hans has seen the car that he wants to buy"				
Main clauses (interrogative)				
Was What	*sieht* sees	*Hans* Hans		
"What does Hans see?"				
	sieht sees	*Hans das Auto* Hans the car		
"Does Hans see the car?"				

Prefield	Left Bracket	Middle Field	Right Bracket	Postfield
Main clauses (imperative)				
	sieh see	*das Auto* the car		
"See the car!"				
Complement clauses				
	dass that	*Hans das Auto* Hans the car	*gesehen hat* seen has	
"that Hans has seen the car"				
Relative clauses				
	das which	*Hans* Hans	*gesehen hat* seen has	
"that Hans has seen"				

Figure 2: Examples of the topological model applied to German sentences

3.2 Separable Prefixes

A large percentage of German verbs fall in the class of separable prefix verbs (in the NLPWIN lexicon, roughly 8,000 of a total of 20,000 verbs fall in this category). The peculiarity of these verbs is that they form a semantic unit, but are separated syntactically into two parts, one of which is a finite verb stem, the other is a prefix that occupies the position of a non-finite verb in the topological model of the sentence. Consider the example *abgeben* which is the German verb meaning "return". This verb consists of two parts, a prefix *ab* and a verb stem *geben*. The semantics is not compositional, although there certainly is at least some overlap between the meaning of the stem *geben* "to give" and the separable prefix verb *abgeben*. In verb-second clauses such as the declarative main clause in Figure 2, the stem and the prefix separate, with the stem occupying the Left Bracket, and the prefix occupying the Right Bracket:

Hans gibt$_{STEM}$ *das Buch ab*$_{PREFIX}$
"Hans returns the book"

The correct positioning of prefix and stem is an integral part of sentence realization in German. Any simple-minded mapping of word-to-word in machine translation, for example, will fail miserably if the target language is German unless some mapping from one verb in English to both a prefix and a stem in German is possible, and their correct positioning in the topological model is ensured.

3.3 Morphological Case

German has a rich system of inflectional morphology. Particularly important for sentence realization as well as parsing in German is case marking on noun phrases. There are four cases in German: nominative, accusative, dative, and genitive. Depending on a number of factors such as the morphological class of the lexical items, number, gender, and the choice of determiner, case can be morphologically realized on various elements of the

noun phrase: the noun itself, and (if present) determiners and adjectives. Case is often an important clue in determining the semantic role of a noun phrase in the clause. If an active clause contains a nominative and an accusative noun phrase, the nominative phrase can safely be assumed to be the subject, and the accusative phrase to be the object, independently of their linear order in the sentence string.

3.4 Constituent Order

The ordering of words and constituents varies across languages, and so does the rigidity with which the canonical order must be obeyed. We will restrict ourselves to the discussion of free constituent order, since neither English nor German can be reasonably claimed to exhibit any free word order in the real sense; i.e., neither English nor German show examples where individual words can be ordered freely, outside of the immediate constituent that they belong to.[2]

English has a relatively rigid constituent order although a number of preposing and extraposing operations can alter that order, so that any simplistic claim about "fixed" constituent order in English is problematic. German, on the other hand, allows many major constituents to be rather freely distributed amongst Prefield and Middle Field, and to a somewhat lesser extent in the Postfield. At the same time, the position of the verb is fixed to the two bracket positions as described in section 3.1. Below are some examples to illustrate this point (English glosses at the bottom of the example):

[Unter diesen Umständen] hat [die Firma] [weitere Lieferungen] bestellt, [ohne abzuwarten].
[Die Firma] hat [unter diesen Umständen] [ohne abzuwarten] [weitere Lieferungen] bestellt
[Weitere Lieferungen] hat [die Firma] [unter diesen Umständen] bestellt, [ohne abzuwarten]
[Ohne abzuwarten] hat [unter diesen Umständen] [die Firma] [weitere Lieferungen] bestellt

Gloss: "Under these circumstances the company has ordered further shipments without waiting."

[ohne abzuwarten] = "without waiting"
[unter diesen Umständen] = "under these circumstances"
[die Firma] = "the company"
[weitere Lieferungen] = "additional shipments"
[... hat ... bestellt] = "has ordered"

All of the above permutations and many more logically possible ones yield grammatical German sentences. At the level of predicate argument structure, the meaning of all the permutations is identical. At finer-grained levels of semantic/ pragmatic description, such as theme/rheme, topic/focus, background/foreground information, there clearly are

[2] We disregard phenomena like floating quantifiers in order to keep the discussion simple.

differences, however. Since none of these finer grained distinctions are currently computable or representable in the NLPWIN framework, we will ignore them for the remainder of this report.

3.5 Extraposition of Clauses

In both German and English, it is possible to extrapose clausal material to the right periphery of the sentence, as the following examples illustrate.

Relative clauses:

English: The man entered the room who usually causes trouble right away.
German: Der Mann hat den Raum betreten, der üblicherweise immer Ärger macht.

Infinitival clauses:

English: The possibility was considered to leave the country.
German: Man hat die Möglichkeit erwogen, das Land zu verlassen.

Complement Clauses:

English: A rumor has been circulating that he is ill.
German: Ein Gerücht ging um, dass er krank ist.

Figure 3 shows that English and German differ in the frequency of this phenomenon. The results shown are based on automatic data profiling with NLPWIN, where the output of the parser has been postprocessed to indicate relative clauses (RELCL), infinitival clauses (INFCL), or complement clauses (COMPCL) that have been moved from their original position. The analysis is based on 100,000 aligned English-German sentence pairs from Microsoft technical manuals and a different German corpus of 62k sentences consisting of a mixture of news, grammar book examples, user input and other sources.

Nearly one third of German relative clauses are extraposed in technical writing, while only 0.18% of English relative clauses are extraposed in the corresponding sentence set. For infinitival clauses and complement clauses the numbers are more comparable between English and German. The high number of extraposed relative clauses in German accords with the number reported in Uszkoreit et al. (1998), who observe 24% of relative clauses being extraposed in a hand-annotated German news corpus.

	German technical corpus (100K sentences)	English technical corpus (100K sentences)	German balanced corpus (62K sentences)
RELCL	32.96%	0.18%	24.06%
INFCL	5.56%	0.82%	4.26%
COMPCL	2.26%	4.61%	2.81%

Figure 3: Percentage of extraposed clauses

Since extraposition is so rare in English, an English sentence realization module could safely ignore extraposition and still result in very fluent output. A complete German sentence realization module, however, will need to model extraposition. The modeling of extraposition is discussed further in Gamon et al. (2002b).

4 The NLPWIN system

4.1 The syntactic analysis: Sketch and Portrait

NLPWIN produces two levels of syntactic output: an initial constituent analysis in which attachment ambiguities are represented in a packed tree (the "Sketch"), followed by a constituent analysis in which attachment ambiguities have been resolved (the "Portrait").

Syntactic trees in NLPWIN are "flattened" representations of a syntactic analysis in terms if binary augmented phrases structure rules (see Jensen et al. 1993 for more details on the formalisms). Each syntactic node in NLPWIN has a head, and it may have pre-modifiers and/or post-modifiers. Figure 4 illustrates a syntactic structure for a German sentence. Nodes in this structure correspond to attribute-value data structures of considerable complexity. Figure 5 shows the record for the node NP2 in Figure 4.

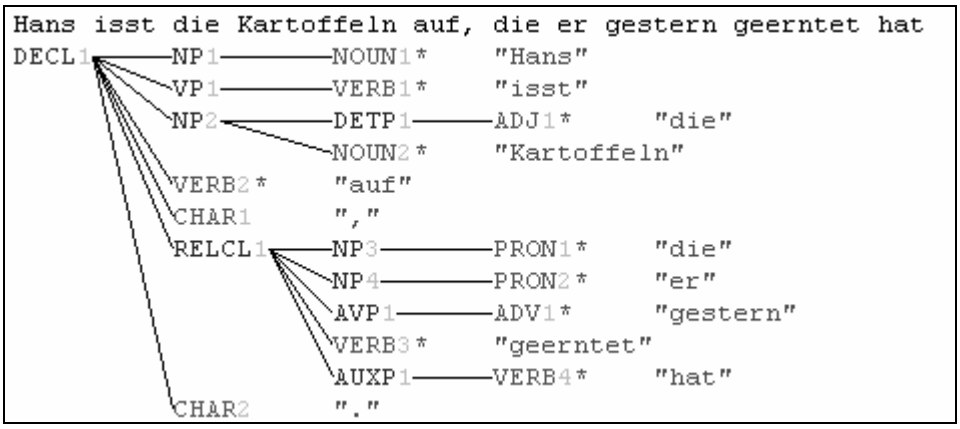


Figure 4: Example of a syntactic structure produced by NLPWIN

```
(Segtype    NP
 Nodetype   NP
 Nodename   NP2
 Ft-Lt      3-4
 String     "die Kartoffeln"
 CopyOf     NP6
 Rules      (NPwDetQuant NOUNtoNP)
 Constits   (AJP1 NP7)
 Lex        "Kartoffeln"
 Lemma      "Kartoffel"
 Bits       Det Art Def Distl
            Prox1 Fem Pers3 Plur
            Closed InitCap LexCap
            Count Pl_nom_str
            Pl_acc_str
 Prmods---DETP1 "die"
 Head-----NOUN2 "Kartoffeln"
 Pod        5
 Prob       0.09594
 Parent---DECL1 "Hans isst die Kartoffeln auf , die er gestern geerntet hat ."
 Preadj---DETP1 "die"
 Mrphrecs
       (Lemma      "Kartoffel"
        Bits       Plur Infld Pl_nom_str
                   Pl_nom_wk Pl_acc_str
                   Pl_acc_wk Pl_dat
                   Pl_gen_str Pl_gen_wk )

       (Lemma      "Kartoffel"
        Bits       Plur Infld Pl_dat )
 CharProb   72 )
```

Figure 5: Example of a syntactic record in NLPWIN

4.2 The semantic representation: Logical Form

Subsequent processing in NLPWIN computes a semantic representation, the "*Logical Form*" (LF). The Logical Form is a graph data structure that represents the core predicate argument structure and basic semantic relations. Semantic relations are encoded as labeled arcs between semantic nodes. Semantic nodes are "lexical" in the sense that they are derived from syntactic nodes and are labeled with the citation form of the head of the syntactic node from which they were derived. There are no abstract semantic nodes in the NLPWIN LF. An example of a logical form graph for the English sentence "You have to click on the tab in order to print the document on the printer" is given in Figure 6.

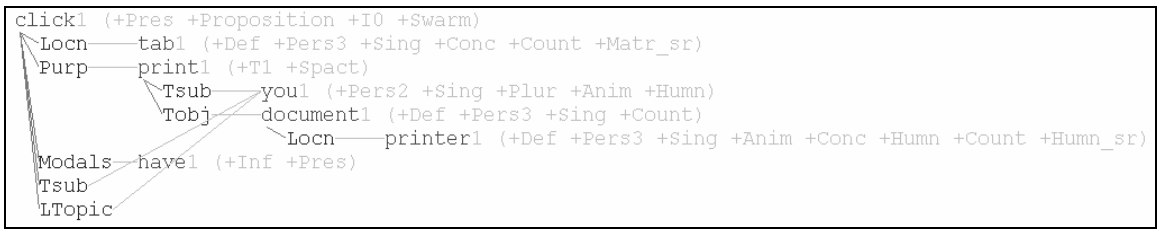


Figure 6: An example of a Logical Form graph

Logical form nodes can carry many features, some of which are related to lexical properties of the associated lemma (e.g., *Conc* on "tab1" in Figure 6), others are semantic features based on the particular analysis (e.g., *Pres* for present tense on the node "click1"). Only content words are represented in logical form, where a content word is understood to be a word that cannot be represented as a small set of features or a label on a semantic arc. The total number of different semantic relations in the NLPWIN system is relatively small; the current system has about 40 semantic relations. Some examples of semantic relations are given below:

- Basic predicate argument structure relations:

  Tsub, Tobj, Tind (for semantic subjects, objects, and indirect objects)

- Other semantic relations:

  Means, Time, Duration, Locn (location), Cause, Mannr (manner), Purp (purpose), Result, Measure, Classifier, Equiv (equivalence), Possr (possessor), LTopic (topic), Props (propositions), Mod (unspecified modifiers)

Semantic nodes in logical form contain pointers to the syntactic nodes they were derived from. This is an important bookkeeping device, enabling us to train models on the correspondences between syntactic and semantic nodes in order to learn the conditions for the operations necessary to transform one into the other.

5 The procedural flow

5.1 The Major Stages of the Amalgam Pipeline

The Amalgam pipeline consists of eight stages, which perform linguistically distinct sets of operations, as shown in Figure 7.

1. Pre-Processing (preparing the LF graph for further processing)
   - Degraphing
   - Addition of lexical information through dictionary lookup
   - Simplification of German compounds
2. Flesh-out (adding syntactic information to the LF graph)
   - Addition of syntactic labels
   - Insertion of function words such as determiners, auxiliaries, semantically empty prepositions, relative pronouns, reflexive pronouns, etc.
   - Assignment of spellout probabilities for NPs in subject or object position
   - Assignment of morphological case features
   - Assignment of verb-position features
3. Conversion to basic tree:
   - Reading off a syntactic tree structure from the degraphed LF
   - Splitting of separable prefixes from their stems
   - Introduction of syntactic representation of coordination
   - Reversing of certain syntactic dominance relations
4. Global movement:
   - Raising, Wh-movement and movement of relative pronouns
   - Extraposition
   - Setting of underspecified verbal inflectional features (agreement, participial features, etc.)
5. Intra-constituent ordering (establishment of linear order)
6. Surface cleanup:
   - Surface realization of determiners, relative pronouns and reflexive pronouns
   - Deletion of duplicated material in coordination (syntactic aggregation)
7. Punctuation
8. Inflectional generation

Figure 7: Overview of the Amalgam pipeline

In general, the order of the stages reflects dependencies in the pipeline. For example, punctuation depends on an established order of constituents, and so does deletion of duplicated material in coordination. Movement, as we treat it in Amalgam, is an operation that moves a constituent out of its parent constituent and attaches it higher in a syntax tree without establishing linear order so the ordering of constituents has to follow hierarchical movement. Assignment of case features, verb-position features, and syntactic labels is best performed before the basic tree is established, in order to provide a complete syntactic representation and helpful information for following stages. Syntactic labels in particular are an important source of information for models downstream in the pipeline.[3]

[3] Clearly, the downstream models could also pick up on the set of features that were predictive of the syntactic label. The use of the syntactic label as input to subsequent models, however, results in more parsimonious decision trees, e.g., for the case assignment model, as experimentation reveals.

A few steps could be performed equally well at different points in the Amalgam pipeline. Function words, for example, are inserted during Flesh-out in Stage 2. We decided that the insertion of function words was conceptually similar to the other operations performed during Flesh-out that augment the Logical Form with syntactic information.

We believe that the order of the major stages is highly language-independent. The order of operations within each stage is not significant.

In section 5.1.1-5.1.8 we will illustrate the workings of these eight stages through screenshots of the corresponding structures/strings on the basis of the example sentence *Hans isst die Kartoffeln auf, die er gestern geernet hat* "Hans eats up the potatoes which he has harvested yesterday". Not all of the individual operations are performed in this one sentence, but it serves as a general illustration of the kinds of operations in the Amalgam pipeline. In sections 6 and 7 we discuss each of the procedural and machine-learned operations in more detail.

For this example, the input LF graph to the generation component is given in Figure 8. To simplify the display of the graph, the display algorithm attempts to minimize crossing lines. The node *Kartoffel1* is displayed twice in the graph but is in fact the same node, i.e. *Kartoffel1* is dependent on *aufessen1* and *ernten1*. Similarly, *Hans1* is dependent on *aufessen1* in the Tsub relation and is a possible intrasentential coreferent of *er1*.

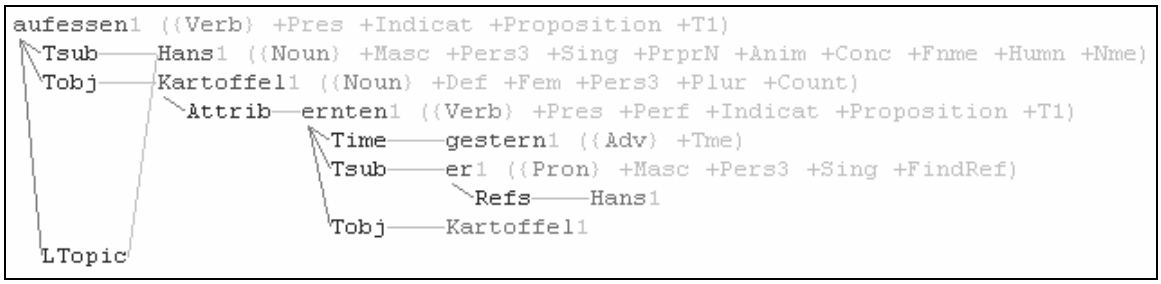


Figure 8: LF structure for the sentence *Hans isst die Kartoffeln auf, die er gestern geernet hat*

5.1.1 Pre-Processing

In the Pre-Processing stage, the LF graph is degraphed; i.e., a structure is created in which each node has at most one parent node. This operation creates a tree structure that facilitates conversion into a syntactic tree in the subsequent stages. Nodes that needed to be duplicated in order to create a tree from the graph bear coindices which link them to their counterparts. In addition, a lexical lookup in the German NLPWIN dictionary is performed on the lexical items present in the graph, and the dictionary information is stored in an attribute on the records in the graph (not displayed in Figure 9). Finally, compound nouns that have been analyzed into components in LF are reconverted into an un-analyzed compound string for training and generation purposes.

The output of the pre-processing stage is illustrated in Figure 9. Note that the nodes *Kartoffel1* and *Hans1* which have previously been in two dependency relations have each been duplicated. *Kartoffel1*, for example, is now only dependent on *aufessen1*. *Kartoffel1* has been cloned to produce *Kartoffel2*, which is dependent on *ernten1*. The coindices are

shown in Figure 9. A positive coindex denotes the node from which another node was cloned. A negative coindex indicates that the node was originally cloned from the node whose coindex is the corresponding positive integer, e.g., *Karotoffel1* (coindex of 2) is the node from which Kartoffel2 (coindex of -2) was derived.

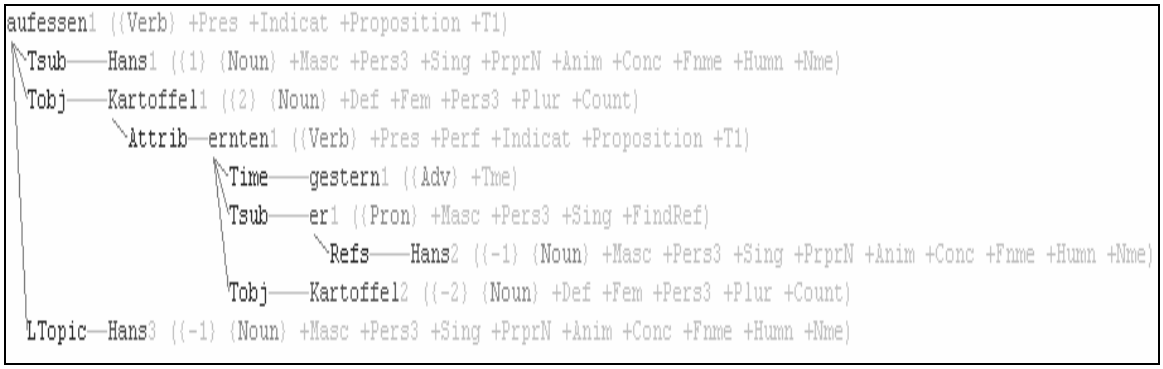


Figure 9: The output of the pre-processing stage

5.1.2 Flesh-out

During the Flesh-out stage, information is added to the degraphed LF. Typically, these are details about syntactic realization that have been normalized at the more abstract LF level of representation. First, syntactic labels are assigned to the nodes in the degraphed LF, based on a decision tree classifier (the new syntactic labels are present in an attribute on the nodes, but are not displayed in Figure 10 below). Function words that carry no semantic information are not present at LF. These function words are inserted next and include:

1. An abstract determiner: “Defdet” for definite determiners, “Indefdet” for indefinite determiners, “Whdet” for Wh determiners, and “Proxldet” and “Distldet” for demonstrative determiners. The surface form of these determiners is determined later, during the “surface cleanup” stage.

2. Auxiliaries.

3. Prepositions which have a purely syntactic function. In German, this includes the prepositions *von* and *durch* used in the passive construction.

4. The infinitival marker *zu*.

5. Negation *nicht*, which is marked in the LF as the feature [+Neg].

6. Subordinating conjunctions *dass* and *ob*.

7. Expletive subjects, i.e. the semantically empty grammatical subject *es*.

8. An abstract relative pronoun “Relpro” which receives its surface realization during stage 6, “surface cleanup”.

9. Reflexive pronouns.

10. The Wh adverbial *wie* “how”.

The function words given in 1-7 are inserted based on a decision tree classifier for each of the insertion tasks. The function words given in 8-10 are inserted by simple functions. In addition to the insertion operations, a function contracts LF nodes of prepositional proforms such as *dadurch*, *damit* ("through that", "with that") etc. to their surface string. Logical subjects and objects are assigned a probability for “spellout” by a decision tree classifier, i.e., a probability of their being realized in the surface string. Logical subjects of infinitival clauses, for example, should not be overtly represented in the string. Finally, case features and verb position features are assigned by decision tree classifiers. Figure 10 shows the result of the flesh-out operations on our sample LF.

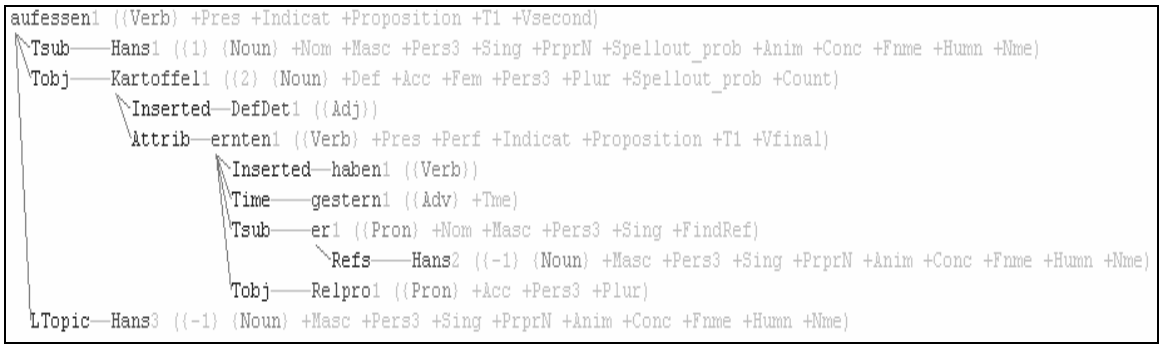


Figure 10: The degraphed LF after Flesh-out

5.1.3 Conversion to basic tree

During conversion to a “basic” tree, the first operation is the actual removal of logical subjects and objects which have a low probability of overt realization (as assigned by a decision tree classifier during Flesh-out). The degraphed LF at this point is transformed into a syntactic tree structure. The syntactic labels on nodes in the degraphed LF that were assigned during Flesh-out are copied over to the corresponding non-terminal nodes in the basic tree. Separable prefixes are split from their stem, based on verb-position features assigned in the previous stage, and based on lexical information about the boundary between the prefix and the stem obtained from the dictionary during Pre-processing. In the next two steps, the representation of coordination is mapped from the way it is handled in LF (see section 4 and section 6.2.9) to a more surface oriented structure in which the coordinating conjunction is the syntactic head. The last step in the conversion to basic tree is an operation based on a decision tree classifier which reverses syntactic dominance relations in those contexts where syntactic and semantic dominance relations are at odds, particularly in cases involving quantification, e.g., *viele der Leute* “many of the people” where *viele* “many” is the syntactic head, but *Leute* is the semantic one. Figure 11 shows the basic tree structure for the example sentence. In parentheses on the far-right, the LF relations of the nodes to their semantic parent are displayed, relations starting with a tilde denote “pseudo-relations”, i.e., inserted material that had no original place in the LF and hence no original LF relation. The presence of the LF relations is a reminder that the new syntactic nodes in the basic tree bear references to the LF nodes

from which they were constructed. The LF features continue to be accessible to downstream modules.

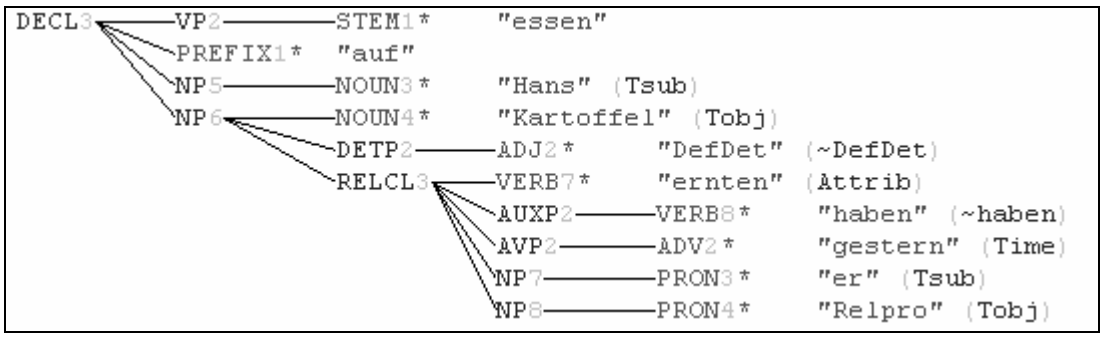


Figure 11: Basic tree structure for the example sentence

5.1.4 Global movement

During Global Movement, non-local movement operations are performed. Non-local here means movement beyond the limits of the immediate parent. All "local" movement in Amalgam is treated as an ordering phenomenon within one constituent, not as genuine movement. Raising, Wh-movement, and the movement of relative pronouns/relative expressions are handled by three simple functions. While this seems strangely at odds with the attention that these operations have received in linguistic research, it is important to note that the more involved examples of multiple Wh-movement, long distance Wh-movement, parasitic gaps, etc., which are important phenomena from a theoretical point of view, are extremely rare in real-life texts. Given the rarity of these phenomena in our training set, we decided to deal with these phenomena with rules. In principle, a machine-learned approach could be applied, given sufficient training examples.

The next movement step, extraposition of relative, infinitival and complement clauses, is based on a decision tree classifier which decides for each instance of such a clause whether it should move up one step and attach to the parent of its parent. Once reattached there, the next "hop" is evaluated, until a position is found where the probability of further movement is less than the probability of no further movement (and hence less than 0.5). A trace is left behind in the original position, with a pointer to the extraposed clause (and vice versa).

The final steps in the global movement stage are two functions which assign morphological features for verbs based on the information present in the tree nodes. Tense information is copied to the finite verb which might be an auxiliary inserted during flesh-out. Participial information is copied onto the non-finite verb, and agreement of the finite verb with the subject is established. The surface subject is identified as the first nominative NP in the domain of the finite verb (case features having been assigned during flesh-out by a decision tree classifier). In our example, as shown in Figure 12, extraposition of the relative clause RELCL3 has taken place.

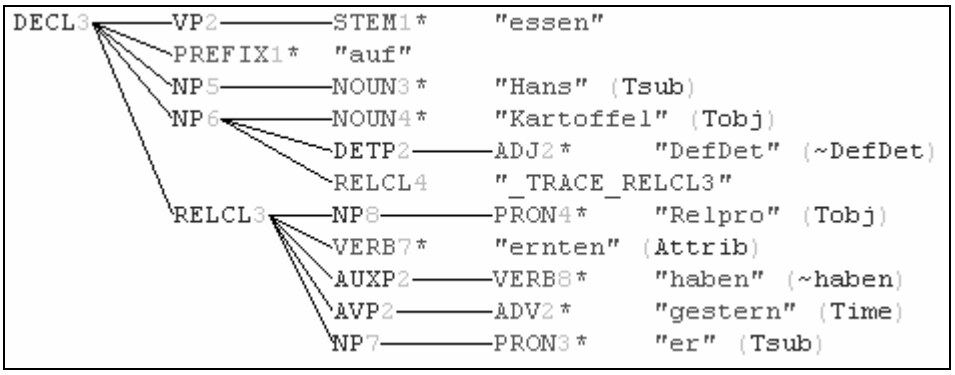


Figure 12: The tree after Global Movement

5.1.5 Intra constituent ordering

During this stage, a generative language model of syntax tree structure is applied in a beam search to establish the linear order of the nodes within each constituent, and consequently the tree. The model consists of n-gram probabilities (currently, n=2) on the order of the labels of the nodes and the semantic relations of the nodes (from the LF), conditioned on constituent features, such as parent and head nodetypes (for more details see section 7.2 below). Currently, we apply a "fix-up" function after the application of the model to correct the positions of the verbs, based on the verb position features assigned by a decision tree classifier in the Flesh-Out stage. We plan to eliminate this function as work on the order model progresses. As currently structured, verb position cannot be reliably established by the order model, due to the limitations of an n-gram window unconditioned on verbal features. For any given n, an n-gram is inadequate for capturing the generalizations about verb position where the finite verb and non-finite verbs in a verb-second structure can be separated by a theoretically unlimited number of constituents. One possible extension to the model that we envision is the addition of a separate model of the verb position.

At the end of the ordering stage we apply a function that assembles compounds from LFs containing nouns with non-possessive noun modifiers. This function is currently only used in machine translation contexts (recall from section 5.1.1 that we simplify compounds when processing German-to-German generation). In our example, ordering results in the tree in Figure 13.

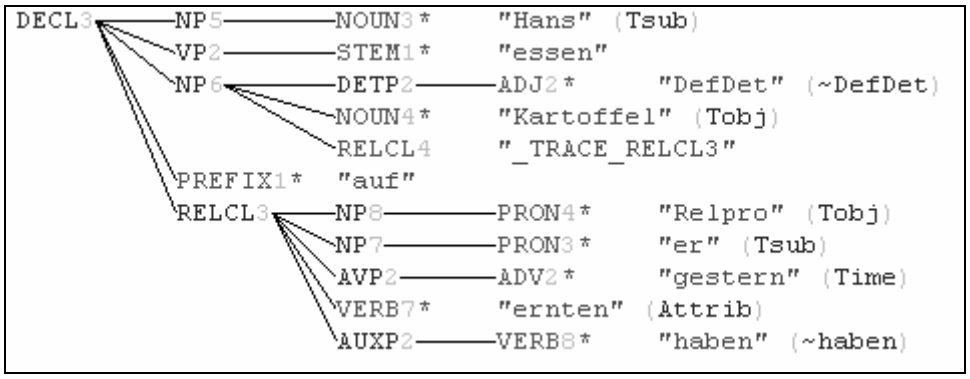


Figure 13: The ordered tree

5.1.6 Surface cleanup

It is obvious from Figure 13 that some work still needs to be done to that structure in order to arrive at a correct surface string. Both the abstract relative pronoun "Relpro" and the abstract determiner "Defdet" need to be converted to their surface realization. These tasks are achieved during surface cleanup by two decision tree classifiers which decide on the most probable surface form. Note that in German this is not a trivial task: during training, the model has picked up on no less than 55 different determiner forms from the training corpus, and 23 different forms of relative pronouns. Reflexive pronouns, which also received an abstract form during insertion in flesh-out, are converted into their surface form by a simple function. The result of these operations for our example sentence is shown in Figure 14.

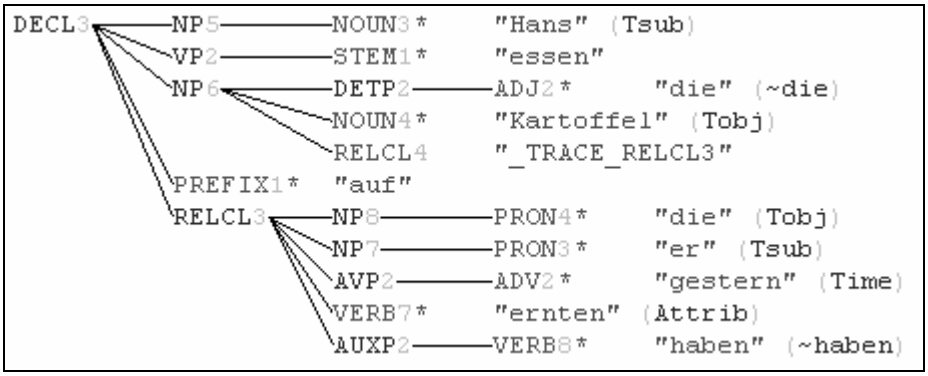


Figure 14: The tree after surface cleanup

Surface cleanup contains an additional step not illustrated in the example sentence: the reduction of duplication in coordinated constituents, also called *syntactic aggregation* in the generation literature. Consider a sentence like *Hans hat die Kartoffeln gekocht und gegessen* "Hans has cooked and eaten the potatoes". The LF for this sentence correctly establishes semantic relations between each of the verbs *kochen* and *essen* and the subject *Hans* and object *die Kartoffeln*. Mapped to a tree through Amalgam, the surface string will faithfully encode all the relations that were present in the input LF, resulting in duplication: *Hans hat die Kartoffeln gekocht und Hans hat die Kartoffeln gegessen* "Hans has cooked the potatoes and Hans has eaten the potatoes". Although this is a perfectly grammatical German sentence, it is not the desired fluent output we would wish to

produce. Surface cleanup contains two operations dealing with syntactic aggregation. The first operation is based on a decision tree classifier which establishes a probability of being overtly realized for each of the duplicated nodes in a coordination structure. Each of the duplicated nodes with p(overtly realized) > 0.5 will be retained, while the duplicated nodes with lower probability are eliminated. In case there is not a single duplicated node that reaches the probability threshold, the node with the highest probability of being realized is retained as a safeguard against truncation. The second operation is a function that eliminates duplicated function words such as prepositions and auxiliaries.

5.1.7 Punctuation

After the creation of an ordered and fully spelled-out tree, punctuation needs to be inserted to ensure fluent and readable output. Punctuation rules are notoriously difficult in German, and although some simplification has been achieved in the spelling reform, there are still 26 different rules for the correct positioning of the comma alone. Since punctuation conventions are typically in the form "insert punctuation X after Y" or "insert punctuation X before Y", we decided to build two different decision tree classifiers for preceding and for following punctuation. We only train and apply these models for sentence internal punctuation, since sentence final punctuation can be inserted with a simple function.[4] At each terminal node in the tree, the left edge of that terminal node and the right edge of the preceding node are passed into the classifier for preceding and following punctuation, respectively. The verdicts from both classifiers are collected and if there is any strong prediction (>0.5) for the insertion of punctuation, the strongest such prediction wins and the predicted punctuation mark is inserted. In our example, one comma is inserted before the extraposed relative clause, as shown in Figure 15.

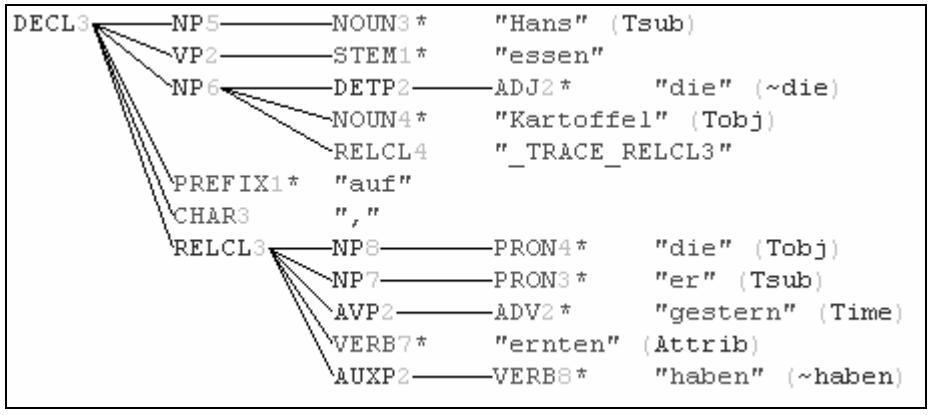


Figure 15: Tree with inserted punctuation

5.1.8 Inflectional generation

The final stage in the Amalgam pipeline is inflectional generation. The records in the tree structure at this stage in the pipeline contain all necessary information to be passed into a

[4] The technical texts do not contain many examples of imperatives with exclamation marks, rhetorical questions, or other cases in which sentences would not end in a period.

rule-based inflectional morphology component for German. This component has been developed for use in the German grammar checker in the Microsoft Word word processor. Features passed into the inflectional generation component include case, gender, number, person, etc. To give an example, the record of the node STEM1 in the tree in Figure 15 is shown in Figure 16. Based on the features Pers2, Sing, Pres, and Indicat, the verb form "isst" can be generated from the Lemma "essen".

```
(Segtype    VERB
 Nodetype   STEM
 Nodename   STEM2
 Ft-Lt      2-2
 String     "isst"
 CopyOf     REC947
 Lex        "isst"
 Lemma      "essen"
 Bits       Pers3 Sing Pres
            Indicat IO T1acc
 Vptc       (mit nach an weg durch ab voll auf aus über hinter satt)
 Infl       Verb-essen
 Parent     VP11 "isst"
 Deriv      (essbar)
 SemNode    aufessen3 )
```

Figure 16: Attribute-value data structure for the finite stem in the example sentence

Figure 17 contains the final result of the generation process on the example sentence, including the inflected forms of the verbs *essen*, *ernten*, and *haben* and of the noun *Kartoffel*.

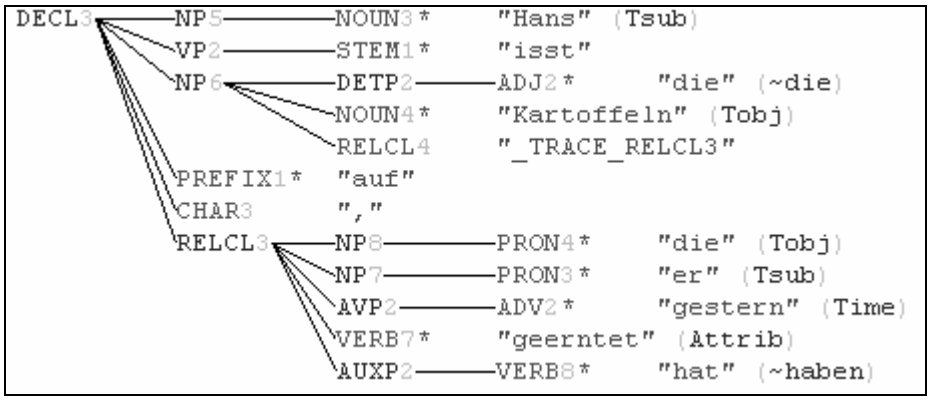
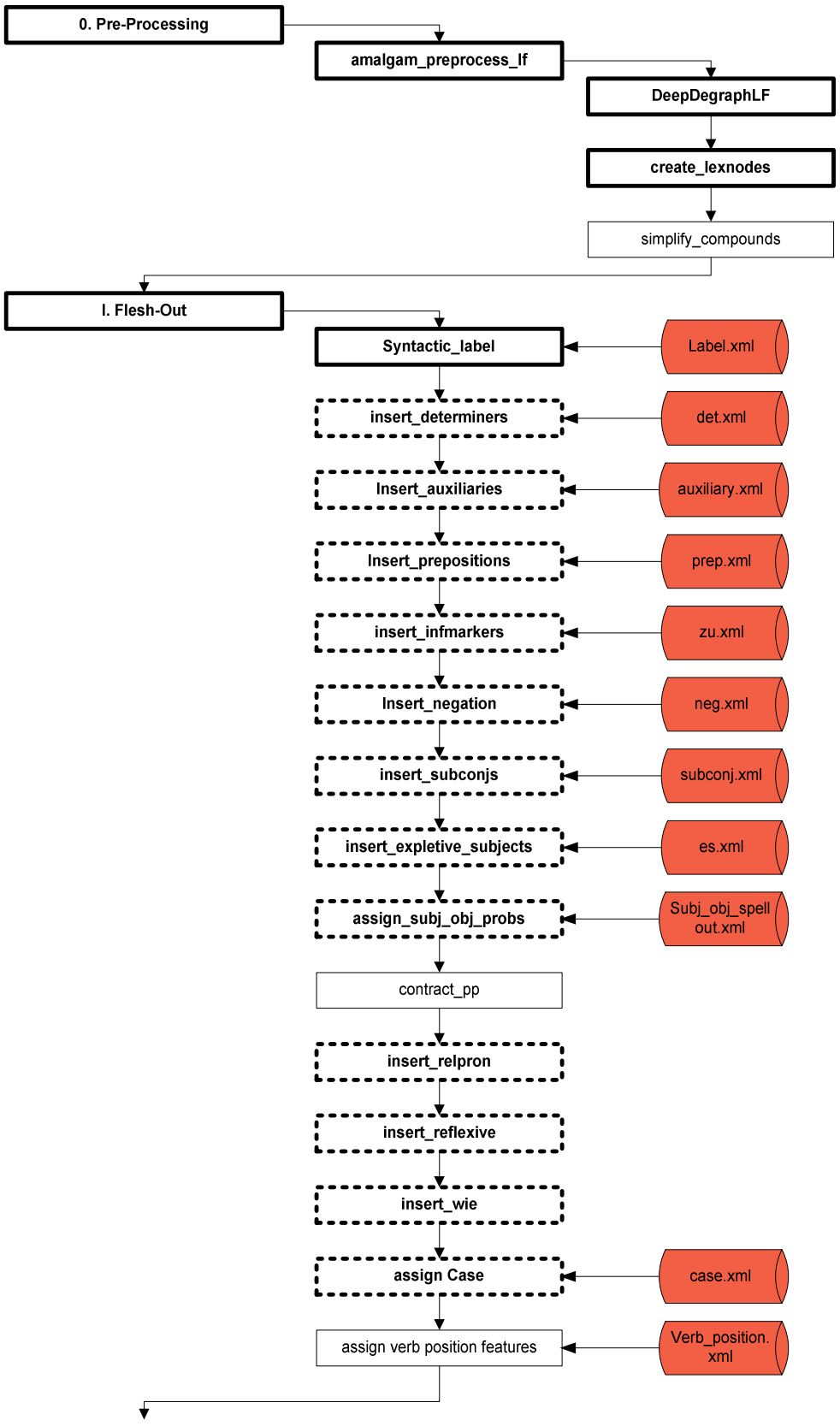
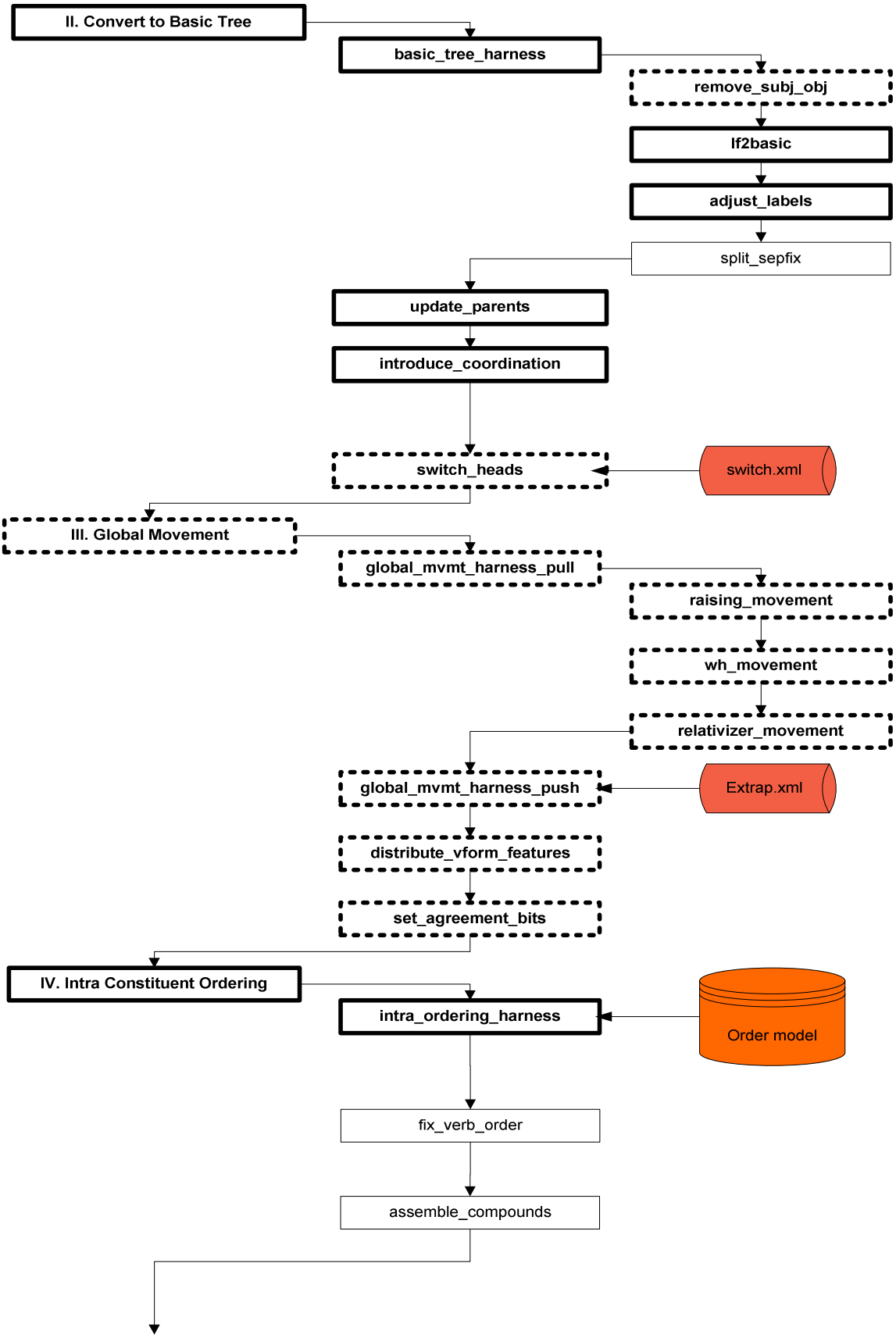


Figure 17: The final inflected tree

A string is read off this final tree structure, and in our case, the output string corresponds exactly to the input string: *Hans isst die Kartoffeln auf, die er gestern geerntet hat.*

5.2 A Detailed Flowchart of the Amalgam Pipeline




II. Convert to Basic Tree
basic_tree_harness
remove_subj_obj
lf2basic
adjust_labels
split_sepfix
update_parents
introduce_coordination
switch_heads
switch.xml
III. Global Movement
global_mvmt_harness_pull
raising_movement
wh_movement
relativizer_movement
global_mvmt_harness_push
Extrap.xml
distribute_vform_features
set_agreement_bits
IV. Intra Constituent Ordering
intra_ordering_harness
Order model
fix_verb_order
assemble_compounds

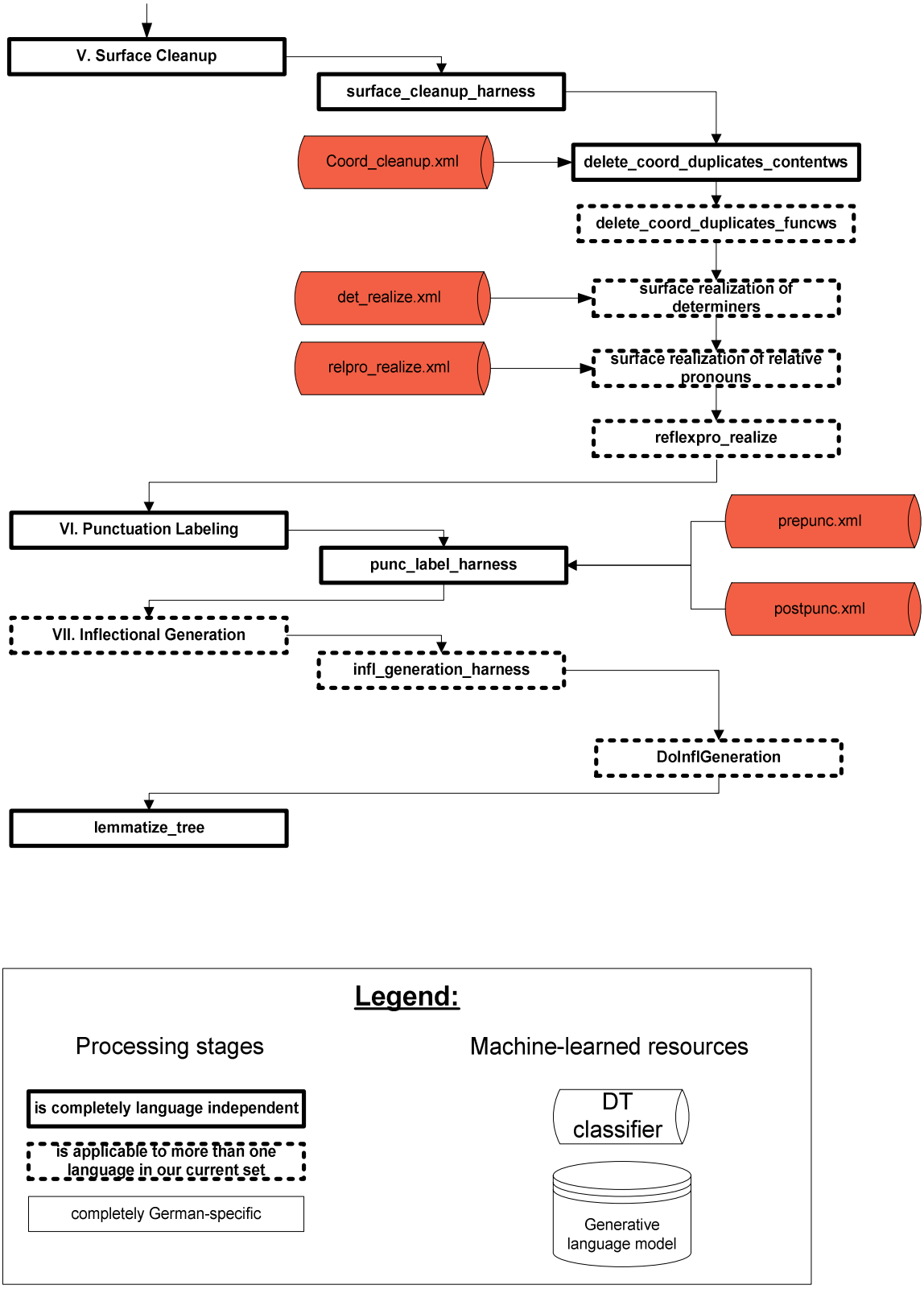


Figure 18: A detailed flowchart of the Amalgam pipeline

6 The rule-based operations in Amalgam

6.1 Degraphing

We begin with a logical form graph as input to the generation process. The first step in producing a linear sequence of words is to disentangle the logical form graph to produce a degraphed logical form.

Each node in the input logical form contains a list of pointers to parent nodes, stored in the attribute *Parents*. A parallel list of atoms, *ParentAttrs*, stores the corresponding labels on the arcs to those parent nodes. These attributes are updated during the degraphing. When the degraphing is complete, every node but the root has exactly one parent. The root has no parent, by definition.

Nodes that are replicated during degraphing are assigned a numerical index, stored in the *CoIndex* attribute. One node is assigned a positive integer, while the duplicates are assigned a negative integer.

During the degraphing operation, certain logical form attributes are ignored (i.e., their values are not cloned). These fall into three classes:

1. System-internal bookkeeping attributes of no linguistic interest, e.g., CopyOf, a pointer to the record that the current record was copied from during construction of the logical form. The other bookkeeping attributes are Originl, Clones, CopyLFCopiedTo, Rules, Constits, CopyOf, BoxCodeChecks, and LexNode.

2. Attributes that used to encode the restructured logical form. These are ParentAttrs, Parents, and CoIndex.

3. Attributes used only for advanced semantic processing, but not yet considered reliable or useful for generation. These include attributes indicating intra-sententential coreference (Refs, RefOf) and the attributes used for MindNet, a semantic knowledge base (Richardson et al. 1998), namely WeightedPaths, MatchPaths, Topicl, Simples, AmbRecs, ExpandSCs, AmbGCs, Counts, Masses, Coordnode, HypSynLems, Emph, and Nominf.

Figure 19 shows the logical form for the sentence *Alle Artikel und Publikationen jeder ausgewählten Datenbank werden übertragen und aktualisiert* “All articles and publications of each selected database are transmitted and updated”. This logical form contains a fair number of nodes with multiple parents, which require degraphing in Amalgam. Figure 20 illustrates the same logical form after degraphing.

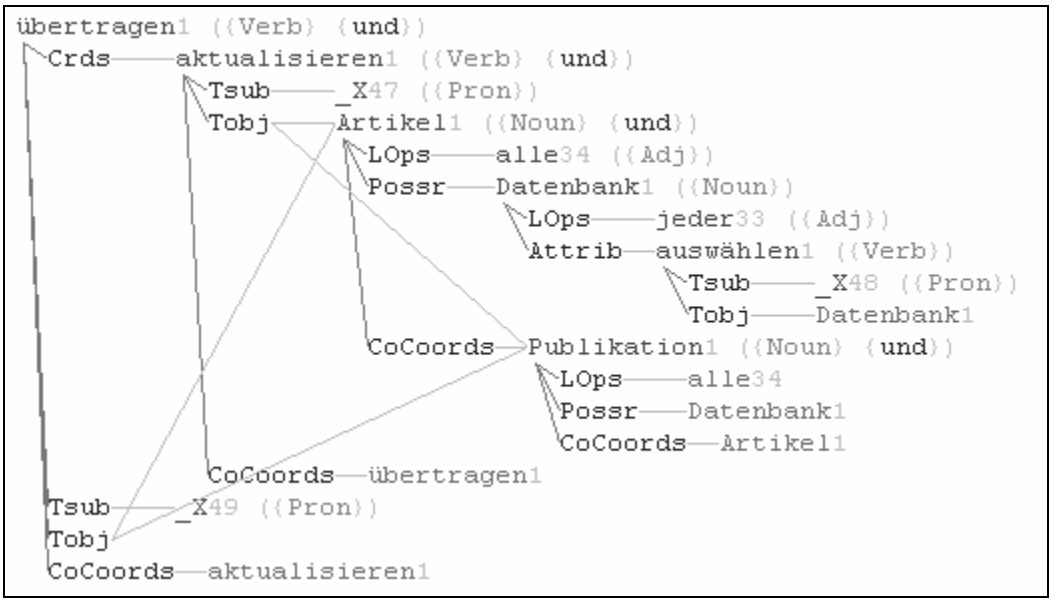


Figure 19: German logical form before degraphing

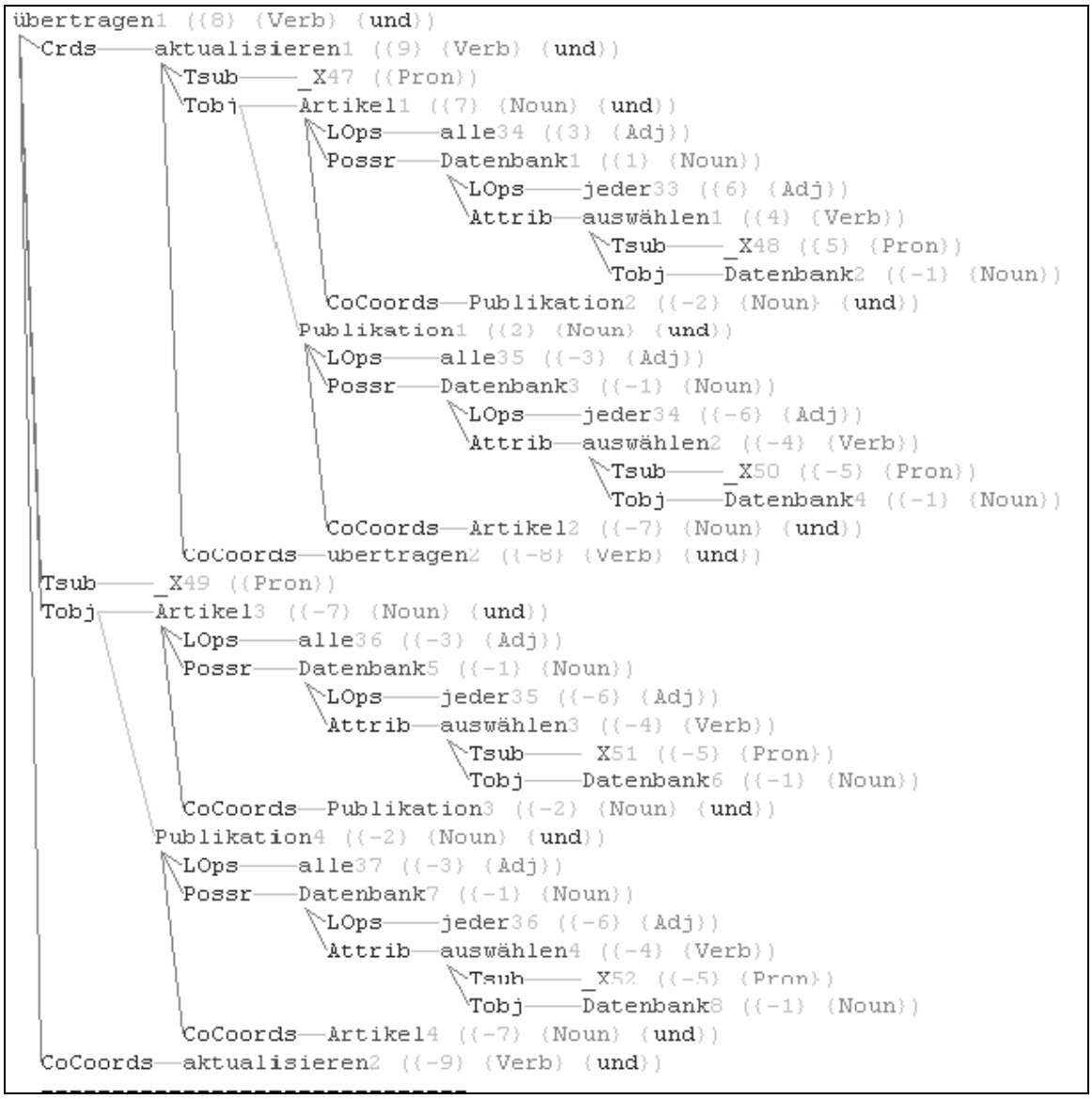


Figure 20: German logical form after degraphing

6.2 Miscellaneous rule-based operations

6.2.1 Creation of lexnodes

The function create_lexnodes performs a lookup of the lemma of a node in the logical form. It then stores the information retrieved from the dictionary in an attribute on that node. The purpose is to make lexical information (such as subcategorization information) available to the subsequent processing stages.

For hyphenated words or compounds, the function performs analysis of the word and returns lexical information of the head of the compound or hyphenated item.

6.2.2 Simplification of compounds

German compounding (especially nominal compounding) is very productive. The German analysis system contains a compounding analysis module, which tries to identify the parts of a compound such as “Eingangsbereich” (entry area) by using lexical information, word frequency information and syllable structure restrictions. The correct analysis in this example is “Eingang” + “s” + “Bereich”, where “s” is what is called a linking morpheme. At the level of logical form, the head of the compound forms a node in the LF graph (in this case “Bereich”), and the other meaning-bearing parts of the compound are linked to that node through the “Mods” attribute. For the purposes of Amalgam training, we ignore the internal structure of compounds, which is part of morphological generation, not syntactic generation. For training purposes, we simplify the logical form representation, effectively undoing the compound analysis. The result of that process on a node of a compound word is a single node with the complete compound string.

6.2.3 Contracting PPs

German has an array of PP proforms such as “damit”, “dafür”, etc. These forms contain a preposition (“mit”, “für” in the example) and a pronominal element “da”. At LF, we currently decompose these words into a representation similar to that of a full PP “mit das” (with that), “für das” (for that). This analysis is of little impact at the moment, but could pave the way for determining the referent of the pronominal part. For the purposes of Amalgam, we simply reconstruct the string of the proforms from the representation at LF, undoing the analysis step.

6.2.4 Insertion of relative pronouns

Relative pronouns in our logical form analyses have been replaced by a copy of the semantic node they are referring to. In order to produce correct output in generation, this copy has to be replaced by a relative pronoun. For an illustration of this, see Figure 9 and Figure 10 above. All the relevant information for this replacement is present during the flesh-out stage: we need to know that the node in question is a copy of a node in the parent chain (this information is encoded in the CoIndex attribute), we need to know that the node in question is inside a relative clause (the label RELCL must have been assigned to the parent), and we need to know that the RELCL in question modifies the original node that the copy was made from (this information is directly encoded as a semantic relation in LF).

6.2.5 Insertion of reflexive pronouns

Reflexive pronouns are used in two contexts in German: there are inherently reflexive verbs, where the reflexive pronoun does not carry any semantic role, and there are normal transitive verbs used reflexively. In the first context, the reflexive does not appear as a node in logical form at all (but the verb is marked with a special feature *ReflexSens*), in the second context, it appears as a copy of the node that it refers to. Insertion of reflexive pronouns picks up on these two different contexts and inserts a reflexive pronoun in the first context, and replaces the copy with a reflexive pronoun in the second context.

6.2.6 Insertion of “wie”

“Wie” is a Wh adverb, like its English counterpart “how”. It is not represented as a node at logical form, since its only function is to carry the Wh feature. Insertion of “wie” is a simple operation that is triggered if there is a Wh feature on a node, but no other Wh-carrying element is present or has been inserted yet.

6.2.7 Converting the fleshed-out LF to a basic tree

This is a recursive transformation of the degraphed LF tree into a syntactic tree structure, such that the LF node label becomes the head of a constituent, and LF modifier nodes become syntactic modifiers. The syntactic labels that have been assigned during the flesh-out stage are copied onto the corresponding nodes in the syntactic tree (by the function adjust_labels).

6.2.8 The splitting of separable prefixes

Splitting a verb into a stem and a separable prefix is triggered when the verb is actually a separable prefix verb (as indicated by a lexical feature) and the verb occurs in a context where the stem should be separated, i.e., either in a verb-initial or in a verb-second structure with no auxiliary or modal verb present that would carry the finiteness features.

If these conditions hold, lexical information on the verb determines where the split between stem and prefix should be made. The node is split into a STEM and a PREFIX node (see Figure 11 for illustration). Verbal inflectional features are copied over to the stem.

6.2.9 Introduction of coordination

Coordination, one of the notoriously difficult aspects of natural language, is represented in different ways at the logical form level and during syntactic analysis. Syntactically, we treat a conjunction as the head of a coordinated construction, with the coordinated phrases and additional conjunctions in the pre- and postmodifiers of that head. Semantically, there is no single node for the coordinated phrase (see Figure 19 for an example). Rather, each of the coordinated phrases has its own node, and enters into semantic relations by itself. In addition, each of the coordinated nodes maintains pointers to the semantic nodes of the other phrases with which it is coordinated in an attribute *CoCoords*. This mismatch in representation is remedied by two functions which adapt the syntactic tree structure that has been built directly from the degraphed logical form. In essence, the functions convert CoCoords into coordinated syntactic nodes, with the conjunction as the head.

6.2.10 Rule-based movement operations

Raising, Wh movement and movement of relative pronouns/expressions are handled in a rule-based manner in Amalgam, as opposed to the extraposition of clauses, which employs a decision tree classifier. This is by no means a necessary design feature, but given the sparsity of data and the simplicity of a rule-based approach, we decided to adopt this strategy in our prototype. We are well aware of the complexity of long distance movement phenomena across languages, and we do not suggest that the simplistic treatment we have chosen is a linguistically adequate solution. However, it is also a fact

that long-distance movement is a rare phenomenon in the data that we currently work with, so that we have no basis in the data for a machine learned approach.

There are currently two raising functions. One function raises nodes out of adjective phrases and noun phrases to the level of the copular verb in predicative contexts, the other function raises subjects of raising verbs. The latter function is a prototype that hasn't been tested, since in the data we don't find enough instances of raising verbs to learn the correct syntactic labels for them. Without correct labeling, this function is not triggered.

Wh movement is triggered if the structure contains a phrase marked by the *Wh* feature that is not dominated by another Wh or WhQ phrase (a direct or indirect Wh question) and if that phrase has an ancestor higher up in the tree that is marked as *WhQ*. Once this context is detected, the Wh phrase is moved up to the WhQ node.

Relativizer movement works very similarly to Wh movement, except that the triggering context here is the presence of a relative pronoun that is not dominated by a relative clause. In this context, the relative pronoun moves up to the first relative clause in its parent chain.

6.2.11 Placement of inflectional features on verbs

Two functions distribute inflectional features to the correct verbal targets. The first function identifies the finite verb (which can be an inserted auxiliary or a modal) and shifts the tense, mood, and finiteness features to that verb. It also marks the non-auxiliary verb as past participle if the construction is marked as perfective or passive, and it marks verbs as infinitives if a modal verb is present and there is no passive or perfective context. The second function identifies the grammatical subject as the first nominative noun phrase that is in the domain of the verb. It then copies person and number features of that noun phrase onto the finite verb. If no grammatical subject is found, a default assignment of 3rd person singular is made.

6.2.12 Fixing up surface order

Given the current limitation of the generative language model employed for ordering, we employ a “fix up” function which deals with the current inability of the order model to account reliably for the position of the verb. This function adjusts the verb positions according to the verb position features that have been assigned by a decision tree classifier in the Flesh-out stage. The function shifts the finite verb to the left bracket position in verb-second and verb-initial structures, and ensures that all non-finite verbs are lined up in the right bracket position. Needless to say, we consider this fix up function a temporary solution that will become obsolete as work on the order model progresses.

6.2.13 Compound generation

Compound nouns are very common in German. In order to provide fluent output, Amalgam needs to be able to provide compounded nouns where the input Logical Form contains a noun modifying another noun directly (i.e. without the presence of a

preposition). This scenario exists especially in the context of machine translation, where input Logical Forms are created from Logical Forms of another language such as English.

Compound generation in Amalgam is rule-based: nominal modifiers on a noun are strung together into one single word string. Information about the linking morpheme (letters that are inserted between nominal parts, depending on lexical information on the left-hand word) is retrieved from the lexicon and taken into account. If no linker information is available in the lexicon, a hyphen is inserted between the parts of the compound as a back-off strategy to facilitate intelligibility.

6.3 Inflectional generation

For inflectional generation, the terminal nodes in the syntactic tree with their inflectional bits and case information (on nouns) are passed into NLPWIN's generation function, which has been developed for the Office grammar checker projects. This generation function utilizes NLPWIN's finite-state morphology, developed with the other analysis components of NLPWIN.

7 The machine-learned components of Amalgam

We provide in-depth descriptions of each of the machine-learned components, beginning with the decision tree classifiers and proceeding to the generative language model of syntactic constituent structure.

7.1 Decision Tree Classifiers

All of the machine-learned modules in Amalgam, with the notable exception of the order model, are based on decision tree classifiers. We use the WinMine toolkit (Chickering, nd.) to build and view our decision trees. For each classification task, we build decision trees at varying levels of granularity (by manipulating the prior probability of tree structures to favor simpler structures) and selected the model with maximal accuracy on the corresponding parameter tuning data set.. Reported accuracy numbers are based on that selected model.

In this section, we discuss each of the decision tree classifiers in turn, providing information on the motivation for the classifier, the input features, and the features actually selected by WinMine. We evaluate the accuracy of each classifier and perform failure analysis.

If not otherwise specified, the decision tree classifiers are built on a set of 100K sentences from the technical domain (computer manuals). The set is split 70/30 for training versus prameter tuning, respectively. We report overall accuracy and baseline accuracy of each model, as well as precision/recall/F-measure for each value of the target feature on the parameter tuning set. The baseline accuracy number is the accuracy resulting from applying the most frequent value of the target feature across the board. Work on the individual models is ongoing, and the numbers reported in this report are from December 2001. Considerable improvements have been made since then. For more recent results, see (Corston-Oliver et al. 2002).

We use standardized sets of features for the training of the classifiers, with special features used sparingly and only with linguistic motivation. To simplify the discussion of the features in the individual sections on decision trees, we will use the following set of terms:

Standard bits:

> An inclusive set of features present on records in NLPWIN. These include subcategorization features, semantic features, tense features, Wh etc. The total number of features in this set is 193. The value of these features is binary: either the bit is present, or it is not.

Standard attributes:

> An inclusive set of attributes present at logical form. The total number of attributes is 33. The value of these features is binary: either the attribute is present, or it is not.

ParentAttrs:

> An attribute denoting the semantic relation of a node to its Parent. The value of this feature is an atom, there are as many values as there are logical form standard attributes.

Cat:

> The part of speech

Crds and CoCoords:

> Coordination-related attributes. *CoCoords* is a list with references to the other nodes with which a semantic node is coordinated. *Crds* has the same function, but is used in root node contexts.

The following is a list of those standard bits that ended up being used in any of the decision tree classifiers:

Person/number bits:

> **Sing**, **Plur**, **Pers3**, **Pers2**

Verb-related bits:

> **Pass** (passive), **Pres** (present), **Indicat** (indicative), **Futr** (future), **Perf** (perfective), **Imper** (imperative), **Condition** (conditional clause), **Modal** (modal verb), **Continuous** (aspectual feature), **Completed** (aspectual feature), **Resultat** (aspectual feature), **Possibl** (modal feature), **YNQ** (yes/no question), **WhQ** (Wh

question), **Etreaux** (verb takes "sein" to form the perfect tense)[5], **I3** (intransitive verb with infinitival clause),

Noun-related bits:

N_ung (noun derived from verb by suffix "ung"), **Fem**, **Masc**, **Neut**, **Def** (definite), **Indef** (indefinite), **Univ** (universal quantification), **Proxl** (demonstrative, indicating something near to the speaker), **Quant** (quantified), **CompPart** (part of a compound), **ExstQuant** (existential quantification), **Wh**, **Reflex** (reflexive), **Rel** (relative),

Subcategorization bits:

I0 (intransitive), **T1** (transitive), **T5** (transitive with that-clause), **T1dat** (transitive with dative object), **T1acc** (transitive with accusative object), **T1gen** (transitive with genitive object), **D1** (ditransitive verb), **D5** (ditransitive verb with that-clause), **L1** (copular verb with NP predicate), **Extrap3** (with infinitival complement that can be extraposed), **V3** (takes an NP and an infinitival complement), **B3** (adjectives/adverbs that take infinitival complements), **F5** (adjectives/adverbs that take that-clause complements), **V2** (takes a bare infinitive as a complement), **T6** (transitive, takes a Wh-clause as complement), **T3** (transitive, takes an infinitival clause as complement), **V2comp** (takes a verb-second complement clause)

Miscellaneous bits:

Neg (negated), **Proposition** (has propositional content), **Time** (time expression)

Standard attributes:

PrpCnjLem (Lemma of prepositional element), **Classifier** (classifier expression), **LOps** (operators), **Modals**, **DegreeMods** (modifiers of degree), **Intnsifs** (intensifiers), **Cause**, **Locn** (location), **Props** (propositions), **SMods** (sentence modifiers), **Time**, **Manner**, **Equiv** (equivalence relation), **Measure**, **Tsub** (subject), **Tobj** (object), **Tind** (indirect object), **Benef** (beneficient), **Matr** (material), **Duration**, **Possr** (possessor), **Mod** (unspecified modifier), **Part** (part relation), **Means**, **Purp** (purpose), **Result**, **Source**, **Goal**, **Attrib** (attributive relation), **LAgent** (agent), **CoAgent**, **PrepRel** (unspecified prepositional relation), **Appostn** (apposition)

A special notation is used in the composition of the feature names to indicate if the feature is tested on the Parents, or Parents of the Parents, and in the case of special features, to indicate whether the feature value is an atom or a continuous value. To give some examples:

[5] Originally used to denote French verbs that form the *passé composé* with the verb *être*.

- 1~I0 = the *I0* bit on the node itself
- 1~I0~Parents = the *I0* bit on the first of the Parents (technically, the Parents attribute is a list, but in a degraphed LF there is (by definition) only one element in that list
- 1~I0~Parents~Parents = the I0 bit on the first of the Parents of the first of the Parents (i.e., on the grandparent)
- A~myfeature = a special feature *myfeature* that has an atom as its value

The 1~ prefix simply indicates that only one record is examined to compute this feature. Since this is the case for all features in Amalgam, it can be ignored.

To give some ballpark figure for the total number of features that are emitted for each of the classification tasks, there are 193 standard bits in addition to the 33 standard attributes. For those models where we test the standard bits and attributes on a node itself, its parent and its grandparent, we emit a total of (193 + 33) * 3 = 678 features.

All the models in the flesh-out stage are trained on (and applied to) logical form nodes, all the models after flesh-out are trained on (and applied to) syntactic nodes in the basic tree. If logical form information needs to be accessed from the basic tree, it is accessed through the SemNode attribute which refers to the corresponding semantic node.

7.1.1 Syntactic labeling

Motivation

In Amalgam, sentence realization is mediated through a syntactic stage. Logical form is converted step by step into a syntactic structure, with the output being very similar to an analysis tree structure. Syntactic labels (especially on non-terminal nodes) are an important part of any syntactic tree, and many linguistic phenomena can be best described with reference to syntactic labels.

Input features

- Cat and ParentAttrs on the node itself, the parent, and the grandparent
- Standard bits and attributes on the node itself, its parent, and grandparent
- Two special features:
  - HasWhDaughter: 1 if the node has a daughter that is marked +Wh, 0 otherwise
  - Pers2~Tsub: the Tsub is Pers2

Features selected

A total of 48 features were selected for this model:

1~Cat, 1~ParentAttrs, 1~PrpCnjLem, 1~Pers3~Parents, 1~Pass~Parents, 1~LOps, 1~ParentAttrs~Parents~Parents, 1~Continuous~Parents, 1~Time~Parents, 1~Pres~Parents, 1~Resultat~Parents, 1~Pers3, 1~CoCoords, 1~Indicat, 1~Cat~Parents, 1~Tsub~Parents, 1~Proposition, 1~Tsub, 1~ParentAttrs~Parents, 1~T1~Parents, 1~LOps~Parents~Parents, 1~Pres, 1~Tobj, 1~Quant~Parents, 1~I0~Parents, 1~CnjLem, 1~Quant, 1~Def, 1~CoCoords~Parents~Parents, 1~PrpCnjLem~Parents, 1~Crds, 1~CoCoords~Parents, 1~Condition, 1~Indef, 1~Pers2~Tsub, 1~Rel, 1~Plur, 1~T1~Parents~Parents, 1~Attrib, 1~Resultat, A~HasWhDaughter, 1~Sing, 1~Completed, 1~Modal~Parents, 1~Proposition~Parents, 1~Quant~Parents~Parents, 1~T5~Parents, 1~Plur~Parents~Parents

Classifier accuracy and complexity

This classifier is built on 10,000 sentences (with a 70/30 split training versus parameter tuning/test). Since the classifier is trained on each semantic node that has a corresponding syntactic node, the number of data points obtained from 10,000 sentences is already very large. The accuracy for this model is 98.27%. The baseline for the model is .35. Precision, recall, and F-measure for each of the values of the target feature are given in Figure 21. The DT classifier has 121 branching nodes.

Key	Precision	Recall	F-measure
DETP	0.9929(140/141)	0.9396(140/149)	0.9655
COMPCL	0.9276(141/152)	0.8545(141/165)	0.8896
VP	0.9807(1928/ 1966)	0.9954(1928/ 1937)	0.9880
QUANP	0.9856(618/627)	0.9968(618/620)	0.9912
AVPNP	0.0000(0/0)	0.0000(0/ 12)	0.0000
IMPR	0.9835(298/303)	1.0000(298/298)	0.9917
AVP	0.9982(1658/ 1661)	0.9846(1658/ 1684)	0.9913
LABEL	0.9296(66/ 71)	0.9041(66/ 73)	0.9167
NAPPOS	0.9838(426/433)	0.9660(426/441)	0.9748
QUES	0.0000(0/0)	0.0000(0/5)	0.0000
AUXP	0.9902(906/915)	1.0000(906/906)	0.9951
NREL	0.7143(30/ 42)	0.5882(30/ 51)	0.6452
AJP	0.9909(3582/ 3615)	0.9942(3582/ 3603)	0.9925
ABBCL	0.0000(0/0)	0.0000(0/8)	0.0000
RELCL	0.9871(686/695)	0.9985(686/687)	0.9928
NP	0.9755(10906/11180)	0.9957(10906/10953)	0.9855
POSS	0.9734(1535/ 1577)	0.9672(1535/ 1587)	0.9703
PRPRTCL	0.0000(0/0)	0.0000(0/2)	0.0000
COMMENT	0.2500(6/ 24)	0.8571(6/7)	0.3871
INFCL	0.9007(136/151)	0.9645(136/141)	0.9315
PP	0.9923(5791/ 5836)	0.9585(5791/ 6042)	0.9751
SUBCL	0.9915(585/590)	0.9701(585/603)	0.9807
DECL	0.9959(1695/ 1702)	0.9953(1695/ 1703)	0.9956
PTPRTCL	0.0000(0/0)	0.0000(0/4)	0.0000

Figure 21: Precision, recall, and F-measure for the syntactic label model

Failure analysis

As is apparent from Figure 21, a few of the syntactic labels used in NLPWIN are very rare in the training data used, so that the model is not able to correctly pick up on the determining factors for these labels. The problematic labels include: AVPNP (noun phrase used adverbially), QUES (question), ABBCL (absolute clause), PRPRTCL (present participle clause), and PTPRTCL (past participle clause).

7.1.2 Determiner insertion

Motivation

Function words that are resolved as features at the level of logical form need to be re-inserted during sentence realization.

Input features

- Nodetype on the node itself, parent and grandparent
- Cat, ParentAttrs on the node itself and the parent
- Cat of the possessor of the node itself
- Standard bits and attributes on the node itself and the parent

Features selected

Fourteen features were selected for this model:

1~Def, 1~Nodetype, 1~Indef, 1~Wh, 1~Proxl, 1~Cat~Possr, 1~Nodetype~Parent, 1~Plur, 1~PrpCnjLem, 1~Sing, 1~Quant, 1~Cat~Parents, 1~Resultat~Parents, 1~Cat

Classifier accuracy and complexity

The classifier accuracy is 97.63%. The baseline is 0.58. Figure 22 shows the numbers for each of the five observed values of the target feature. The determiner insertion model has nineteen branching nodes.

Key	Precision	Recall	F-measure
NoDet	0.9905 (11284/11392)	0.9719 (11284/11610)	0.9811
Whdet	1.0000 (22/22)	0.9565 (22/23)	0.9778
Proxldet	0.9922 (508/512)	0.9203 (508/552)	0.9549
DefDet	0.9433 (6068/6433)	0.9859 (6068/6155)	0.9641
Indefdet	1.0000 (1765/1765)	0.9893 (1765/1784)	0.9946

Figure 22: Precision, recall, and F-measure for the determiner insertion model

Failure analysis

The majority of incorrect classifications in this model stems from coordinated noun phrases, where (at least for German) the determiner is often only spelled out once.

Feature extraction for this model should be refined, so that either coordination is taken into account, or coordinated NPs are consistently ignored during feature extraction.

7.1.3 Auxiliary insertion

Motivation

Function words that are resolved as features at the level of logical form need to be re-inserted during sentence realization.

Input features

- ParentAttrs and Cat on the node itself
- Standard attributes and bits on the node itself and the parent
- Crds and CoCoords on the node itself and the parent
- Standard bits on the LexNode of the node itself
- Etreaux bit on the LexNode of the node itself

Features selected

Thirteen features are selected.

1~Pass, 1~Perf, 1~Completed, 1~Proposition, 1~Pres, 1~Condition, 1~CnjLem, 1~Etreaux~LexNode, 1~Past, 1~D1~LexNode, 1~T1, 1~ParentAttrs, 1~Tsub

Classifier accuracy and complexity

The accuracy of this classifier is 99.86%. The baseline is 81.36%. The classifier has 14 branching nodes.

Key	Precision	Recall	F-measure
sein_werden	1.0000(59/ 59)	0.6413(59/ 92)	0.7815
sein	0.9800(147/150)	0.8698(147/169)	0.9216
haben	0.9713(744/766)	0.9960(744/747)	0.9835
werden	0.9967(15223/15274)	0.9969(15223/15271)	0.9968
none	0.9993(71048/71098)	0.9997(71048/71068)	0.9995

Figure 23: Precision, recall, and F-measure for the auxiliary insertion model

Failure analysis

Recall is somewhat poor with the combination of the two auxiliaries “sein” and “werden” (in passive perfective). Although we have not performed any detailed failure analysis, it seems to be no accident that the combination of “sein” and “werden” is also the rarest in the data.

7.1.4 Preposition insertion

Motivation

In German, the prepositions "von" and "durch" in passive contexts are semantically vacuous and are not represented at the level of logical form. During sentence realization, they need to be inserted under the appropriate circumstances.

Input features

- Standard bits and attributes on the node itself and the parent
- Standard bits on the grandparent
- ParentAttrs and Cat on the node itself
- Lexical bits (subcategorization and nominal derivational bits) on the LexNode of the parent and the grandparent

Features selected

Nineteen features were selected.

1~Tsub~Parents, 1~Pass~Parents, 1~N_ung~LexNode~Parents, 1~Continuous~Parents, 1~Cat, 1~Sing, 1~ParentAttrs, 1~T1~Parents, 1~Def, 1~Indef, 1~Proposition, 1~Attrib, 1~Quant, 1~Tobj~Parents, 1~Possr, 1~Tobj, 1~Plur~Parents~Parents, 1~Proposition~Parents, 1~Pers2

Classifier accuracy and complexity

Accuracy is 99.15%. The baseline was 97.04%. The model has 20 branching nodes. This model was trained on 10k sentences, since each nominal node was taken into consideration, which yielded a large number of data points.

Key	Precision	Recall	F-measure
von	0.8559(202/236)	0.7953(202/254)	0.8245
none	0.9951(10502/10554)	0.9996(10502/10506)	0.9973
durch	0.7500(27/ 36)	0.4091(27/ 66)	0.5294

Figure 24: Precision, recall and F-measure of the preposition insertion model

Failure analysis

It is not surprising that it is not easy for the model to predict the choice of prepositions "von" and "durch". The choice of prepositions in the German passive is governed by intricate semantic details in interpretation, which generally go beyond the level of granularity of our logical form representation. A cursory failure analysis on the model confirmed that the most important factor in mis-classifications is indeed the distinction between those two prepositions.

7.1.5 Insertion of infinitival marker

Motivation

As with other function words like auxiliaries and determiners, the infinitival marker is semantically vacuous and therefore is not represented as a semantic node at the level of logical form. During sentence realization, it needs to be inserted under the appropriate circumstances.

Input features

- ParentAttrs, Cat and Nodetype on the node itself
- Standard bits and attributes on the node itself and on the parent
- Crds and CoCoords on the node itself and the parent
- Standard bits on the grandparent

Features selected

Fourteen of the input features were selected:

1~Nodetype, 1~Pres, 1~PrpCnjLem, 1~I0, 1~Tobj, 1~Tsub~Parents, 1~Modal, 1~ParentAttrs, 1~V2~Parents, 1~PrpCnjLem~Parents, 1~CnjLem, 1~Past, 1~T1, 1~CnjLem~Parents

Classifier accuracy and complexity

The accuracy of the classifier is 99.77%. The baseline was 95.66%. The decision tree has 15 branching nodes. The model was trained on 10k sentences since every verbal node was a data point.

Key	Precision	Recall	F-measure
zu	0.9634(342/355)	0.9856(342/347)	0.9744
none	0.9993(7630/ 7635)	0.9983(7630/ 7643)	0.9988

Figure 25: Precision, recall, and F-measure for the classifier for insertion of infinitival markers

Failure analysis

Cursory failure inspection reveals that faulty parses (yielding erroneous LFs) are an important factor in mis-classifications. Once the overall analysis of the sentence is incorrect, proper identification and marking of infinitivals becomes very difficult.

7.1.6 Negation insertion

Motivation

Negation is represented at the level of logical form as a feature *Neg*. The decision as to where to insert negation during sentence realization is not a completely straightforward

one, though, because during analysis, the *Neg* feature can percolate up in the tree under various circumstances.

Input features

- ParentAttrs on the node itself
- Cat and Nodetype on the parent and grandparent
- Standard attributes and bits on the node itself and the parent
- Standard bits on the grandparent
- Crds and CoCoords on the node itself, parent, and grandparent
- Two special features:
    - Negquant: is 1 if any of the descendants has the *ExstQuant* bit, indicating that it is in the scope of a negative operator
    - NegquantSister: is 1 if any of the sister nodes has a descendant which bears the *ExstQuant* bit

Features selected

Fifty-eight features were selected:

1~Cat,1~Neg, 1~Nodetype, 1~ParentAttrs, F~NegquantSister, 1~Nodetype~Parents, F~Negquant, 1~Sing, 1~Mod, 1~Pass~Parents, 1~Quant, 1~PrepRel, 1~Tobj, 1~Cat~Parents, 1~Nodetype~Parents~Parents, 1~BndPrp, 1~Tsub, 1~Time, 1~Mod~Parents, 1~SMods, 1~Pass, 1~CompPart, 1~Pres, 1~Cat~Parents~Parents, 1~Modals~Parents, 1~Plur, 1~Pers3, 1~ExstQuant, 1~Proxl, 1~Resultat~Parents, 1~Modal, 1~PrepRel~Parents, 1~LAgent~Parents, 1~Proposition, 1~Tsub~Parents, 1~I0, 1~Plur~Parents, 1~Sing~Parents, 1~Indef, 1~CoCoords~Parents, 1~Tobj~Parents, 1~I0~Parents, 1~Def, 1~Condition, 1~Indicat~Parents, 1~Pres~Parents, 1~Univ, 1~Indicat, 1~T1~Parents~Parents, 1~Proposition~Parents, 1~Condition~Parents, 1~Sing~Parents~Parents, 1~Modal~Parents, 1~Def~Parents, 1~Plur~Parents~Parents, 1~Modals, 1~Indef~Parents, 1~Quant~Parents

Classifier accuracy and complexity

The accuracy is 90.94% with a baseline of 80.79%. The resulting model has 138 branching nodes.

Key	Precision	Recall	F-measure
Insert_neg	0.7969(2648/ 3323)	0.7058(2648/ 3752)	0.7486
none	0.9319(15100/16204)	0.9572(15100/15775)	0.9444

Figure 26: Precision, recall, and F-measure for insertion of negation

Failure analysis

It is not surprising that for each given node, the decision of whether negation should be overtly realized at that very position is far from trivial. Since we know from the properties of our logical forms that each *Neg* feature corresponds to an overtly realized negation at that node or any descendant of that node, we apply the negation insertion model in the following way: If a *Neg* feature is encountered at node X, a probability for insertion of negation is assigned by the model for X and all its descendants. Whichever node receives the highest probability for insertion will be the target for insertion.

7.1.7 Insertion of subordinating conjunctions

Motivation

The subordinating conjunctions "dass" and "ob" are not represented at logical form as nodes, and hence need to be inserted during sentence realization.

Input features

- Standard attributes and bits on the node itself, the parent, and the grandparent
- ParentAttrs, Cat and Nodetype on the node itself
- Subcategorization bits on the LexNode of the parent and the grandparent

Features selected

Twenty-two features were selected:

1~Proposition,1~ParentAttrs,1~Pres~Parents~Parents,1~Pers3~Parents~Parents,1~Proposition~Parents~Parents,1~Equiv,1~Proposition~Parents,1~Pers3~Parents,1~T1acc~LexNode~Parents,1~V2comp~LexNode~Parents,1~Mod,1~Tobj,1~Pres~Parents,1~Pass~Parents,1~D5~Parents,1~T1~Parents,1~L1,1~Pass,1~I0~Parents,1~T1,
1~T1acc~LexNode~Parents~Parents, 1~Plur~Parents

Classifier accuracy and complexity

The accuracy of the classifier for the insertion of subordinating conjunctions is 95.47%. The baseline is 54.55%. The model contains 27 branching nodes.

Key	Precision	Recall	F-measure
dass	0.95(1251/ 1323)	0.92(1251/ 1353)	0.93
none	0.95(2003/ 2106)	0.97(2003/ 2075)	0.96
ob	1.00(432/432)	1.00(432/433)	1.00

Figure 27: Precision, recall and F-measure for the classifier for insertion of subordinating conjunctions

Failure analysis

It is to be expected that the insertion of "ob" is very reliable, given the nature of our logical form representation, where "whether" type clauses are marked with the *YNQ*

feature. The insertion of "dass" is less straightforward in German: a subset of verbs which take complement clauses actually allow complementizer-less complement clauses (with the same verb position as main clauses). For these verbs, then, there is a genuine choice between a complement clause with "dass" and one without. The numbers in Figure 27 bear out that prediction.

7.1.8 Insertion of expletive subjects

Motivation

Expletive (or pleonastic) subjects are semantically empty subjects that are necessary on purely grammatical grounds. They serve as placeholders for other constituents that have been displaced from the subject position. In some constructions, the insertion of expletive subjects is the only option, for example in the existential construction in German: "es gibt verschiedene Möglichkeiten" (*there are different possibilities*), which cannot be expressed grammatically without the presence of the "es" subject. By definition, a semantic representation such a the logical form in NLPWIN will not encode purely grammatical markers such as the expletive subject "es" in German. This makes it necessary to decide where to insert expletive subjects during sentence realization.

Input features

- ParentAttrs, Cat and Nodetype on the node itself
- Standard bits and attributes on the node itself, and the parent
- Standard bits on the grandparent
- Standard bits and attributes on the Tobj (semantic object)
- Subcategorization bits on the LexNode of the node itself, the parent, the grandparent, and the Tobj
- Special features:
    - Cat of the Tobj
    - "Geben": i.e. is the Lemma of the node "geben" or not (this feature allows the model to zero in on the existential construction)

Features selected

Twenty-seven features are selected:

1~B3~LexNode~Tobj, geben, 1~Tsub~Tobj, 1~ParentAttrs, 1~Nodetype, 1~I3~LexNode, 1~PrepRel, 1~Indicat, 1~T1acc~LexNode~Parents~Parents, 1~Def~Tobj, 1~T1acc~LexNode, 1~I0~LexNode, 1~F5~LexNode~Tobj, 1~V3~LexNode, 1~T3~LexNode, 1~Extrap3~LexNode~Tobj, 1~Possibl, 1~Pers3~Parents~Parents, 1~Sing~Tobj, 1~T1~LexNode, 1~T1dat~LexNode, 1~Modals, 1~L1~LexNode, 1~PrepRel~Parents, 1~T1acc~LexNode~Parents, 1~Proposition, 1~Props~Tobj, 1~Mod

Classifier accuracy and complexity

The accuracy is 99.69%, with a baseline of 99.0%. The classifier has 8 branching nodes.

Key	Precision	Recall	F-measure
no	0.9971(79613/79846)	0.9998(79613/79632)	0.9984
yes	0.9116(196/215)	0.4569(196/429)	0.6087

Figure 28: Precision, recall and F-measure for the expletive subject insertion model

Failure analysis

It is obvious from Figure 28 that the recall for insertion of "es" with a value of about 0.4569 is currently still problematic. Since the precision is rather high, there is indication that some grammatical contexts for the insertion of "es" are missed, something that will require more detailed failure analysis.

7.1.9 Assignment of probabilities for the spellout of NPs

Motivation

Semantic nodes at the level of logical form are not always overtly realized in the sentence string. The prototypical examples are the subjects of infinitival clauses, which are logically present and part of the interpretation of the sentence, but are not part of the surface string. In our logical form representation, there is a variety of other scenarios where arguments of predicates are present as semantic nodes (and linked to other semantic nodes), but are not present in the corresponding surface string. Our strategy for dealing with these phenomena is to employ a decision tree classifier which will produce a probability for surface realization for any given (subject or object) node. These probabilities are then stored in an attribute on the node in question. During the conversion to the basic tree, within a set of subject/object nodes that are related to each other, the probability values are examined. All nodes with a probability of overt realization below 0.5 are deleted from the tree, with the safeguard that among each set at least one node must be realized (to avoid truncation).

Input features

- ParentAttrs on the node itself and the parent
- Cat on the node itself
- Nodetype of the parent
- Standard bits on the node itself, the parent, and the grandparent
- Standard attributes on the parent

Features selected

Fifty-six features are selected for this model:

1~Nodetype~Parents, 1~Sing, 1~ParentAttrs~Parents, 1~Pres~Parents, 1~Cat, 1~Pers2, 1~Plur, 1~Def, 1~Proxl, 1~Pass~Parents, 1~Indef, 1~Proposition~Parents, 1~I0~Parents, 1~CompPart, 1~Quant, 1~Pers3, 1~T1~Parents, 1~Imper~Parents, 1~Continuous~Parents, 1~Modal~Parents, 1~Plur~Parents~Parents, 1~T5~Parents, 1~T1~Parents~Parents, 1~Sing~Parents~Parents, 1~Condition~Parents, 1~Indicat~Parents, 1~Indef~Parents~Parents, 1~Perf~Parents, 1~BndPrp~Parents~Parents, 1~Proposition, 1~Def~Parents, 1~L1~Parents~Parents, 1~L1~Parents, 1~Past~Parents, 1~Pers3~Parents, 1~Imper~Parents~Parents, 1~Pers3~Parents~Parents, 1~Univ, 1~I0~Parents~Parents, 1~CnjLem~Parents, 1~Modal~Parents~Parents, 1~Pass~Parents~Parents, 1~Sing~Parents, 1~Proposition~Parents~Parents, 1~Def~Parents~Parents, 1~D1~Parents~Parents, 1~Indicat~Parents~Parents, 1~Condition~Parents~Parents, 1~V2~Parents~Parents, 1~Pres~Parents~Parents, 1~CompPart~Parents, 1~T3~Parents~Parents, 1~CnjLem~Parents~Parents, 1~Futr~Parents, 1~Plur~Parents, 1~V3~Parents~Parents

Classifier accuracy and complexity

The accuracy is 88.59%. The baseline is 68.19%. There are 447 branching nodes in the decision tree, making it the most complex classifier in Amalgam.

Key	Precision	Recall	F-measure
no	0.8951 (23473/26224)	0.7263 (23473/32319)	0.8019
yes	0.8827 (66540/75386)	0.9603 (66540/69291)	0.9198

Figure 29: Precision, recall and F-measure for the subject/object realization model

Failure analysis

This model is one of the most complex in Amalgam, indicating that the task at hand is very difficult. While the precision and recall numbers are satisfactory, there is still room for improvement. One possible strategy is to train the model on non-fitted parses only (i.e. only on those parses that have a spanning analysis for the whole input string), to have it focus on true grammatical generalizations, and not be distracted by faulty parses and noise in the input.

7.1.10 Assignment of Case

Motivation

Case is an important feature in the German grammar. Recall from Section 3.3 that there are four different cases in German (accusative, nominative, dative, and genitive). Constituent order is relatively free in German, and often only the case-marking on a noun phrase will indicate whether it is to be interpreted as the subject, object, or indirect object in a sentence. During sentence realization, case serves as a proxy for grammatical subjecthood etc. For surface realization it is therefore imperative to identify the case of a given noun phrase properly, in order to produce intelligible output.

Input features

- ParentAttrs on the node itself, the parent and the grandparent

- Nodetype of the node itself, the parent and the grandparent
- Pred (logical form equivalent of lemma) on the parent
- Cat on the node itself, the parent and grandparent
- Standard bits on the node itself (with the exception of person and number bits)
- Standard bits on the parent and grandparent
- Standard attributes on the node itself, the parent and the grandparent
- Lexical subcategorization bits on the LexNode (containing the information after lexical lookup) of the parent and the grandparent
- Five special features:
    - Parent_lemma_geben: 1 if the lemma of the parent is “geben”
    - Prep_lemma: the lemma of the governing preposition
    - PrepandPrepdat: 1 if there is a governing preposition and it is marked as *Prepdat* (governs the dative)
    - PrepandPrepacc: 1 if there is a governing preposition and it is marked as *Prepacc* (governs the accusative)
    - PrepandPrepgen: 1 if there is a governing preposition and it is marked as *Prepgen* (governs the genitive)

Features selected

Seventy-two features were selected by the model building process:

A~PrepandPrepdat, 1~Pred~Parents, 1~ParentAttrs, 1~Nodetype, A~Prep_lemma,
A~PrepandPrepacc, 1~Pass~Parents, A~PrepandPrepgen, 1~CoCoords,
1~Nodetype~Parents, 1~Neg, 1~Tobj, 1~T1~Parents, 1~Nodetype~Parents~Parents,
1~Indef, 1~Tind~Parents~Parents, 1~ParentAttrs~Parents, 1~Continuous~Parents,
1~BndPrp, 1~Attrib, 1~Resultat~Parents, 1~T1gen~LexNode~Parents,
1~D1~LexNode~Parents, A~Parent_lemma_geben, 1~X7~Parents~Parents,
1~ParentAttrs~Parents~Parents, 1~I0~Parents, 1~Def, 1~Imper~Parents~Parents,
1~CompPart, 1~Cat~Parents, 1~L1~Parents, 1~Sing~Parents, 1~Tobj~Parents~Parents,
1~T5~Parents, 1~Sing~Parents~Parents, 1~Plur~Parents~Parents, 1~Tind~Parents,
1~CoCoords~Parents~Parents, 1~Quant, 1~Past~Parents, 1~Mod, 1~Indef~Parents,
1~Possr, 1~Cat, 1~T1dat~LexNode~Parents~Parents, 1~Proposition~Parents, 1~LOps
1~Plur~Parents, 1~Proposition, 1~Adjdat~LexNode~Parents, 1~T1~Parents~Parents,
1~Modals~Parents, 1~PrepRel~Parents, 1~I0~Parents~Parents, 1~Tobj~Parents,
1~T1dat~LexNode~Parents, 1~Def~Parents, 1~PrepRel, 1~Completed~Parents,
1~PrpCnjLem~Parents, 1~Pass~Parents~Parents, 1~Cat~Parents~Parents,

1~PrepRel~Parents~Parents, 1~Condition~Parents, 1~T1acc~LexNode~Parents
1~Indicat~Parents, 1~Tsub~Parents~Parents, 1~Pres~Parents,
1~T1acc~LexNode~Parents~Parents, 1~Pers3~Parents~Parents, 1~Pres~Parents~Parents

Classifier accuracy and complexity

Accuracy is 96.02%. The baseline is 0.46. The model has 226 branching nodes.

Key	Precision	Recall	F-measure
Dat	0.9562 (17575/18380)	0.9797 (17575/17940)	0.9678
Acc	0.9257 (5132/5544)	0.8783 (5132/5843)	0.9014
Gen	0.9883 (9950/10068)	0.9796 (9950/10157)	0.9839
Nom	0.9563 (4576/4785)	0.9460 (4576/4837)	0.9512

Figure 30: Precision, recall and F-measure for the case model

Failure analysis

In German, it is often very difficult during analysis to determine exactly which noun phrases belong together and which do not. Since any number of noun phrases can be strung in a sequence in the middle field of the German sentence, misanalyses are common especially with out-of-vocabulary nouns. Not surprisingly, "stranded" or misanalyzed noun phrases cannot be reliably assigned case, accounting for many of the errors that the case model makes.

7.1.11 Assignment of verb position features

Motivation

As discussed at length in section 3.1, one of the most important aspects of constituent order in German is the correct positioning of the verb. Our strategy in Amalgam is to use a decision tree classifier to assign features that indicate the verb-positioning pattern in the constituent. Downstream models and functions (such as the order model, the order fix-up functions, syntactic aggregation etc.), can then utilize the information present in these features.

Input features

- ParentAttrs on the node itself
- Nodetype of the node itself, the parent, and the grandparent
- Standard bits on the node itself, the parent, and the grandparent
- Standard attributes on the parent, the parent, and the grandparent
- Lexical subcategorization bits on the parent, and the grandparent
- Two special features:

- A~NTlabelCoordmother: the Nodetype of the parent if the node is coordinated

- A~EmptySubject: 1 if the subject lemma is "_X" (indicating an empty, uncontrolled subject)

Features selected

Forty-one features were selected:

1~ParentAttrs, 1~Nodetype, A~EmptySubject, 1~Imper, 1~Tobj, 1~Props~Parents, 1~Proposition, 1~Modals, 1~Indicat, 1~T3, 1~CoCoords, 1~T5~Parents, 1~T5, 1~PrepRel, 1~T6, 1~Tsub, 1~Pass, 1~T1, 1~Mod, 1~I0, 1~PrpCnjLem, 1~Modal, 1~Neg, 1~Proposition~Parents, 1~Condition, 1~Perf, 1~CnjLem, 1~V2comp~LexNode~Parents, 1~L1, 1~Mod~Parents, 1~Pres~Parents, 1~Def~Parents, 1~Tind, 1~Props, 1~Nodetype~Parents, 1~Time, 1~Modal~Parents, 1~WhQ, 1~Modals~Parents, 1~YNQ, 1~Tsub~Parents

Classifier accuracy and complexity

The accuracy is 94.66%, with a baseline of 0.42. The resulting model has 115 branching nodes.

Key	Precision	Recall	F-measure
initial	0.9650 (7107/7365)	0.9818 (7107/7239)	0.9733
final	0.9374 (16669/17782)	0.9750 (16669/17097)	0.9558
undefined	0.5946 (776/1305)	0.3673 (776/2113)	0.4541
second	0.9721 (18637/19172)	0.9719 (18637/19175)	0.9720

Figure 31: Precision, recall and F-measure of the verb position model

Failure analysis

It is reassuring that the determination of verb-second, verb-final, and verb-initial patterns can be made at a fairly high level of precision and recall. There are, however, quite a few cases where the verb position is determined to be "undefined". A legitimate example of indeterminacy of verb position is a simple construct of the form "er geht" (he goes). This verb phrase can be either a verb-second verb phrase (as witnessed by the fact that it can stand alone as a declarative sentence), and a verb-final verb phrase (if, for example, preceded by a subordinating conjunction which requires verb-final position as in "dass er geht" (that he goes). For these "undefined" cases, the only way to determine the correct verb-position pattern is by linguistic generalization as in "this verb phrase is used as a declarative sentence, so in this case it has to be a verb-second structure according to the grammar of German". Since this line of reasoning is not applicable by a purely data-driven model, we will always have a set of data that will escape complete classification into a verb-position pattern.

7.1.12 Inversion of dominance

Motivation

There are grammatical constructions where syntactic dominance is the opposite of semantic dominance. For German as we treat it in NLPWIN, the only example is a quantified expression of the form "viele der Leute" (many of the people), where "viele" is the syntactic head of the noun phrase, but semantically it is "Leute" which is the head, with NLPWIN "viele" as an operator modifying it. While this phenomenon is rather limited in German, we found that in the NLPWIN analysis of French, modal verbs are treated as syntactic heads (because modals in French pattern syntactically with main verbs rather than with non-modal auxiliary verbs), with a reversal of dominance at the level of logical form. In order to prepare for porting Amalgam to French, we decided to face this issue right away, even though the effect in German is very limited due to data sparsity, and very straightforward since there is only one particular construction which exhibits this property. The approach we have taken is to learn in a decision tree classifier the circumstances under which the reversal of dominance between syntactic and semantic relations takes place, and then perform a reversing operation in the basic tree accordingly.

Input features

- Nodetype and Cat of node itself and parent
- Standard attributes and bits on the node itself, the parent, and the grandparent

Note that the switch model and all following models are trained on, and operate on the basic tree, not the logical form. Only the flesh-out models are applied to logical form structures. Therefore, features of the form *1~X~SemNode* mean the presence of feature X on the logical form node (SemNode) of the current node, whereas features of the form 1~X mean the presence of X on the current node.

Features selected

For German, only two features were selected:

1~Nodetype~SemNode, 1~LOps~Parents~Parents~SemNode

Classifier accuracy and complexity

The accuracy is 99.69%. The baseline is 92.0%. The model for German is exceedingly simple, with only 3 branching nodes.

Key	Precision	Recall	F-measure
no	0.9998(5974/ 5975)	0.9968(5974/ 5993)	0.9983
yes	0.9645(516/ 535)	0.9981(516/ 517)	0.9810

Figure 32: Precision, recall and F-measure for the dominance switching model

Failure analysis

As stated above, this model is of an exploratory nature, in preparation for other languages. It is not surprising that for the one simple construction in German that exhibits dominance switching, it is easy to learn the triggering factors reliably.

7.1.13 Extraposition

Motivation

See the detailed discussion of the importance of extraposition phenomena in German in section 3.5 and the discussion in Gamon et al. (2002b). Our approach to extraposition is to determine for each extraposable node (INFCL, COMPCL, RELCL) whether the node should move up one step from its current attachment (its parent node) to the next higher node (its grandparent). From the new position, another assessment is made for the next possible movement step and so on. For training, we extract the value "Yes" for the target feature for each intermediate node between the source position of a clause and its extraposed position - meaning that the answer to the question "should the clause move up one step?" is "Yes". Similarly, we extract the value "No" for the extraposed position (since the clause obviously has not moved higher from there). In the case of non-extraposed clauses we extract a "No" for the parent node of the clause, meaning that there is no movement from the current position[6].

Input features

- Nodetype of cargo node
- Nodetype of the potential origination node for movement, of its parent, and its grandparent
- ParentAttrs of the potential origination node for movement, of its parent, and its grandparent
- Vfinal feature on the potential origination node for movement, on its parent, and its grandparent
- Vsecond feature on the potential origination node for movement, on its parent, and its grandparent
- HasSepfix (indicating a separable prefix verb) on the potential origination node for movement, on its parent, and its grandparent
- Standard attributes and bits on the the potential origination node for movement, on its parent, and its grandparent

[6] We have also explored a different strategy for modeling extraposition: instead of "successive cyclic" one-step movements, it is possible to ask for each ancestor node of the parent of the cargo node whether it is a suitable target or not. At the current stage, both approaches yield comparable results. For a more detailed discussion of the two different strategies, see Gamon et al. (2002b).

- Standard attributes and bits on the cargo node (the RELCL/INFCL/COMPCL that could potentially be extraposed), its parent, and grandparent
- Nodetype of the cargo node
- ParentAttrs on the cargo node
- Six special features zeroing in on the relevant aspects of verb position and "heaviness" as triggers for extraposition:
    - F~NumTokens: number of tokens of the cargo node
    - F~SentenceLengthInToken: length of whole sentence in tokens
    - F~NumChars: number of characters of the cargo node
    - F~SentenceLengthInChar: length of whole sentence in characters
    - A~inVfinalVP: "yes" if any ancestor of the cargo node is a Vfinal VP, "no" otherwise
    - A~inVsecondVP: "yes" if any ancestor of the cargo node is a Vsecond VP, "no" otherwise

Features selected

Sixty features were selected during the model building process (features with the "1~" prefix are extracted on the node under consideration for the "movement one step up yes or no" classification, features with the "2~" prefix are extracted on the extraposable node):

1~Tsub~SemNode, 1~HasSepfix~LexNode~SemNode~Parent, 1~ParentAttrs~SemNode,
A~inVfinalVP, 1~Modals~SemNode~Parent,
1~HasSepfix~LexNode~SemNode~Parent~Parent, 1~Pass~SemNode~Parent,
1~Pass~SemNode~Parent~Parent, 1~Modals~SemNode~Parent~Parent,
F~SentenceLengthInChar, 1~ParentAttrs~SemNode~Parent~Parent,
2~Pass~SemNode~Parent~Parent, 2~Proposition~SemNode, 1~Nodetype~Parent,
1~Pers3~SemNode~Parent, 1~Vsecond~Parent, 2~Modals~SemNode, 1~Nodetype,
2~T1~SemNode~Parent~Parent, 1~Mod~SemNode~Parent,
1~PrepRel~SemNode~Parent, A~inVsecondVP, 1~PrepRel~SemNode,
1~Tobj~SemNode~Parent, 1~Vfinal, 1~Tsub~SemNode~Parent,
2~Tsub~SemNode~Parent~Parent, 2~Attrib~SemNode~Parent, F~NumChars,
1~BndPrp~SemNode, 1~Nodetype~Parent~Parent, 1~ParentAttrs~SemNode~Parent,
2~Mod~SemNode~Parent~Parent, 2~Indicat~SemNode~Parent~Parent, F~NumTokens,
2~ParentAttrs~SemNode, 2~Nodetype, 1~T1~SemNode~Parent~Parent,
2~Proposition~SemNode~Parent~Parent, 2~CoCoords~SemNode~Parent,
2~Indef~SemNode~Parent 1~Def~SemNode, 1~PrepRel~SemNode~Parent~Parent,
2~PrepRel~SemNode, 1~Mod~SemNode~Parent~Parent, 1~Pers3~SemNode,
1~PrpCnjLem~SemNode~Parent, 1~Plur~SemNode~Parent,
2~PrepRel~SemNode~Parent~Parent, 1~Sing~SemNode~Parent,

2~Def~SemNode~Parent, 1~Plur~SemNode~Parent~Parent,
2~Tobj~SemNode~Parent~Parent, 2~PrpCnjLem~SemNode~Parent, 2~T1~SemNode,
1~Sing~SemNode, 2~I0~SemNode~Parent~Parent, 2~Pers3~SemNode~Parent,
2~Plur~SemNode~Parent, 1~Plur~SemNode

Classifier accuracy and complexity

The accuracy is 88.16% with a baseline of 0.67. The model has 116 branching nodes.

Key	Precision	Recall	F-measure
No	0.9181 (4720/5141)	0.9051 (4720/5215)	0.9115
Yes	0.8094 (2102/2597)	0.8331 (2102/2523)	0.8211

Figure 33: Precision and recall of the extraposition model

Failure analysis

Extraposition is a non-trivial linguistic phenomenon with a complicated array of triggering factors, including some not completely understood notion of "heaviness" (where in general a "heavier" clause tends to extrapose more easily than a "lighter" clause - see the discussion in Uszkoreit et al 1998). Additionally, the failure to extrapose, or extraposition in a situation where there should not be any often results not in ungrammatical sentences, but in sentences with varying degrees of unnaturalness and lack of fluency. Detailed error analysis in this context would benefit greatly from correlation with human judgements, an experiment that we have not yet undertaken.

7.1.14 Realization of determiners

Motivation

There are 55 different determiner forms observed in the training data. While the realization of the determiner could be determined by rule, we decided to train a decision tree qualifier for the task.

Input features

- Lemma
- ParentAttrs on the SemNode
- Gender and number bits on the parent
- Case on the parent
- Standard attributes on the parent

Features selected

Eighteen features were selected.

F~ParentCase, 1~ParentAttrs~SemNode~Parent, 1~Lemma, 1~Fem~Parent,
1~Masc~Parent, 1~Neut~Parent, 1~Plur~Parent, 1~ParentAttrs~SemNode~Parent~Parent,

1~Sing~Parent, 1~Mod~SemNode~Parent, 1~Tobj~SemNode~Parent,
1~LOps~SemNode~Parent, 1~PrepRel~SemNode~Parent,
1~PrpCnjLem~SemNode~Parent, 1~Attrib~SemNode~Parent,
1~Appostn~SemNode~Parent, 1~Possr~SemNode~Parent, 1~Tsub~SemNode~Parent

Classifier accuracy and complexity

The accuracy is 90.77% with a baseline of 21.65%. The resulting decision tree classifier has 266 branching nodes.

Key	Precision	Recall	F-measure
derjenigen	0.0000(0/ 0)	0.0000(0/ 3)	0.0000
denselben	0.0000(0/ 0)	0.0000(0/ 45)	0.0000
meiner	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
dieselben	0.0000(0/ 0)	0.0000(0/ 40)	0.0000
ein_und_derselbe	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
meines	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
desselben	0.0000(0/ 0)	0.0000(0/ 13)	0.0000
sein	0.3333(12/ 36)	0.8000(12/ 15)	0.4706
derselbe	0.0000(0/ 0)	0.0000(0/ 4)	0.0000
das	0.9148(3531/ 3860)	0.8768(3531/ 4027)	0.8954
die	0.9432(13327/ 14129)	0.9091(13327/ 14659)	0.9259
dem	0.9003(7402/ 8222)	0.9426(7402/ 7853)	0.9209
den	0.8894(4320/ 4857)	0.8171(4320/ 5287)	0.8517
einem	0.8134(1539/ 1892)	0.8927(1539/ 1724)	0.8512
einen	0.8409(1438/ 1710)	0.8489(1438/ 1694)	0.8449
der	0.9042(14292/ 15807)	0.9504(14292/ 15038)	0.9267
des	0.9771(3378/ 3457)	0.9740(3378/ 3468)	0.9756
dasselbe	0.0000(0/ 0)	0.0000(0/ 17)	0.0000
einer	0.8931(1954/ 2188)	0.8890(1954/ 2198)	0.8910
eines	0.9089(1067/ 1174)	0.9744(1067/ 1095)	0.9405
dies	0.0000(0/ 0)	0.0000(0/ 3)	0.0000
diesem	0.8551(655/ 766)	0.9590(655/ 683)	0.9041
diesen	0.7409(266/ 359)	0.8837(266/ 301)	0.8061
demselben	0.0000(0/ 0)	0.0000(0/ 63)	0.0000
denjenigen	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
derselben	0.0000(0/ 0)	0.0000(0/ 57)	0.0000
meine	0.0000(0/ 0)	0.0000(0/ 7)	0.0000
dieser	0.7526(791/ 1051)	0.9295(791/ 851)	0.8318
dieses	0.9296(581/ 625)	0.9222(581/ 630)	0.9259
diese	0.9057(1248/ 1378)	0.8820(1248/ 1415)	0.8937
seinem	0.2857(6/ 21)	1.0000(6/ 6)	0.4444
welche	0.9118(124/ 136)	0.8671(124/ 143)	0.8889
seinen	0.0000(0/ 0)	0.0000(0/ 12)	0.0000
diejenige	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
ihr	0.8039(123/ 153)	0.8913(123/ 138)	0.8454

Key	Precision	Recall	F-measure
seiner	0.5313(17/ 32)	0.7083(17/ 24)	0.6071
seines	0.0000(0/ 0)	0.0000(0/ 9)	0.0000
ihrem	0.8974(175/ 195)	0.8663(175/ 202)	0.8816
ihren	0.7818(86/ 110)	0.7748(86/ 111)	0.7783
welchem	0.5833(14/ 24)	0.9333(14/ 15)	0.7179
welchen	0.0000(0/ 0)	0.0000(0/ 15)	0.0000
ihrer	0.8296(112/ 135)	0.8682(112/ 129)	0.8485
mein	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
ihres	0.8082(59/ 73)	0.9077(59/ 65)	0.8551
welcher	0.4627(31/ 67)	0.9118(31/ 34)	0.6139
ihre	0.9012(155/ 172)	0.8031(155/ 193)	0.8493
welches	0.0000(0/ 0)	0.0000(0/ 20)	0.0000
dieselbe	0.0000(0/ 0)	0.0000(0/ 41)	0.0000
ein	0.8845(2511/ 2839)	0.9003(2511/ 2789)	0.8923
ein_und_demselben	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
diejenigen	0.0000(0/ 0)	0.0000(0/ 7)	0.0000
eine	0.9606(3823/ 3980)	0.8926(3823/ 4283)	0.9253
ein_und_derselben	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
meinem	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
seine	0.8148(22/ 27)	0.5946(22/ 37)	0.6875

Figure 34: Precision, recall and F-measure for determiner realization

Failure analysis

Figure 34 shows that not all determiners can be realized with the same accuracy. While the numbers are high for the common determiners such as the definite (der/die/das) and the indefinite determiner (ein/eine), the numbers degrade the rarer the form of the determiner is. It is worth noting that there is a fair amount of linguistic ambiguity: some nouns have more than one gender (stemming from different senses), some nouns can be both singular and plural. Additionally, the level of granularity of the features used to describe the semantic import of determiners at logical form is not sufficient in all cases to uniquely determine the particular lexical choice of determiner.

7.1.15 Realization of relative pronouns

Motivation

Twenty-three different forms of relative pronouns are present in the training set. As with the realization of determiners, we have chosen to build a model for the correct choice, instead of hand-crafting a selection process.

Input features

- Lemma of the grandparent and the great-grandparent
- Case
- Nodetype
- ParentAttrs on the SemNode and on the SemNode of the grandparent
- Gender and number features on the grandparent and the great-grandparent
- Standard attributes on the grandparent and great-grandparent

Features selected

Nineteen variables were selected:

F~RelproCase, 1~Masc~Parent~Parent, 1~Fem~Parent~Parent, 1~Plur~Parent~Parent, 1~Nodetype, 1~Neut~Parent~Parent, 1~Sing~Parent~Parent, 1~Lemma~Parent~Parent, 1~ParentAttrs~SemNode, 1~ParentAttrs~SemNode~Parent~Parent, 1~CoCoords~SemNode~Parent~Parent, 1~Fem~Parent~Parent~Parent, 1~Masc~Parent~Parent~Parent, 1~Sing~Parent~Parent~Parent, 1~Possr~SemNode~Parent~Parent, 1~Plur~Parent~Parent~Parent, 1~Lemma~Parent~Parent~Parent, 1~Attrib~SemNode~Parent~Parent~Parent, 1~Tobj~SemNode~Parent~Parent

Classifier accuracy and complexity

The accuracy is 87.79%. The baseline is 53.59%. There are 77 branching nodes in the decision tree classifier.

Key	Precision	Recall	F-measure
wozu	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
dessen	1.0000(46/ 46)	0.8214(46/ 56)	0.9020
die/der	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
was	1.0000(10/ 10)	0.4167(10/ 24)	0.5882
welche	0.0000(0/ 0)	0.0000(0/ 13)	0.0000
wo	0.7000(7/ 10)	1.0000(7/ 7)	0.8235
deren	0.9406(95/ 101)	0.9896(95/ 96)	0.9645
derer	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
das	0.9202(369/ 401)	0.7953(369/ 464)	0.8532
dem	0.8122(480/ 591)	0.8743(480/ 549)	0.8421
die	0.9225(3248/ 3521)	0.9425(3248/ 3446)	0.9324
womit	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
den	0.8875(142/ 160)	0.6544(142/ 217)	0.7533
der	0.8400(882/ 1050)	0.7854(882/ 1123)	0.8118
woraus	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
welcher	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
welches	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
woher	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
warum	0.0000(0/ 0)	0.0000(0/ 2)	0.0000
wobei	0.6907(67/ 97)	1.0000(67/ 67)	0.8171
das/der	0.0000(0/ 0)	0.0000(0/ 1)	0.0000
denen	0.6727(298/ 443)	0.9113(298/ 327)	0.7740
wodurch	0.0000(0/ 0)	0.0000(0/ 26)	0.0000

Figure 35: Precision, recall and F-measure for the realization of relative pronouns

Failure analysis

Similar to the situation with the realization of determiners, it is mostly the rarer forms of relative pronouns that cannot be accurately determined, due to data sparsity. The same caveats with respect to linguistic ambiguity hold for both determiners and relative pronouns.

7.1.16 Syntactic aggregation

Motivation

Any semantic representation of coordination has to encode more than may be present in the syntactic structure. Consider a simple sentence such as “John cooked and ate the steak”. Semantically, there are two conjoined propositions, “John cooked the steak” and “John ate the steak”. There is nothing grammatically wrong with spelling out those two conjoined propositions in the lengthy form “John cooked the steak, and John ate the steak”. Natural language, however, tends to employ strategies to reduce redundant material in coordination by deleting some of the duplicates. This is generally viewed as a sub-area of aggregation in the generation literature (Wilkinson 1995, Shaw 1998, Reape and Mellish 1999, Dalianis and Hovy 1993). In Amalgam, we only deal with sentence-based realization tasks currently, so the approach we take is strictly intra-sentential, along

the lines of what has been called *conjunction reduction* in the linguistic literature (McCawley 1988). While this may seem a fairly straightforward task compared to inter-sentential, semantic and lexical aggregation, it should be noted that the cross-linguistic complexity of the phenomenon makes it much less trivial than a first glance at English would suggest. In German, for example, position of the verb in the coordinated VPs plays an important role in determining which duplicated constituent can be omitted.

In Amalgam, we try to arrive at a reasonable level of fluency in our output, which makes it necessary to model these reduction phenomena in coordination. The model is trained on and applied to the logical form representation that corresponds to the syntax tree.

Input features

- Cat of the node itself, the parent and the grandparent
- ParentAttrs of the node itself, the parent and the grandparent
- Nodetype of the node itself, the parent and the grandparent
- Standard bits and attributes on the node itself, the parent and the grandparent
- Vsecond and Vfinal features on the node itself, the parent and the grandparent
- Two special features:
    - F~HeadMod: indicates whether the node in question is a premodifier or a postmodifier of the head of its parent
    - F~AllVerbpos: indicates for VP-coordination if all coordinated VPs are Vfinal or Vsecond

Features selected

Fifteen features are selected during the construction of the model:

A~HeadMod, 1~Proposition~Parents, A~AllVerbpos, 1~Tobj~Parents,
1~Nodetype~Parents, 1~Nodetype, 1~ParentAttrs, 1~ParentAttrs~Parents, 1~CoCoords,
1~T1~Parents, 1~ParentAttrs~Parents~Parents, 1~Cat, 1~Plur~Parents,
1~Proposition~Parents~Parents, 1~CoCoords~Parents~Parents

Classifier accuracy and complexity

The accuracy is 96.93%, with a baseline of 0.85. The resulting model has 21 branching nodes. The values of the target feature are *last*, *first*, and *middle*: *last* indicates that the node in question is spelled out in the last of the coordinated constituents, *first* indicates that it is spelled out in the first of the coordinated constituents, and *middle* indicates that it is spelled out in a coordinated constituent that is neither first nor last.

Key	Precision	Recall	F-measure
Last	0.9164 (986/1076)	0.8851 (986/1114)	0.9005
First	0.9786 (6022/6154)	0.9854 (6022/6111)	0.9820
Middle	0.0000 (0/0)	0.0000 (0/5)	0.0000

Figure 36: Precision, recall and F-measure for the syntactic aggregation model

Failure analysis

The data confirm the initial linguistic hypothesis that coordination reduction is a matter of spelling out a constituent either at the beginning or at the end of coordination, but not somewhere in the middle.

Cursory error inspection shows that most of the misclassifications seem to result from either bad analyses, or situations where the verb position has not been uniformly determined as Vsecond or Vfinal in all coordinated VPs.

7.1.17 Punctuation

Motivation

Clearly, punctuation is different from the previously discussed phenomena: it is not a core linguistic phenomenon, but rather a matter of orthographic convention. There are two reasons, however, why we believe that punctuation should be part of Amalgam:

- without appropriate punctuation, the output of generation - especially for real-life, complex sentences - is difficult to read and hard to parse for a human consumer.

- even though punctuation is an orthographic convention, it is based on linguistic structure: most punctuation rules make reference to constituenthood, types of constituents etc.

Based on the observation that most (if not all) punctuation conventions which we are aware of are of the form “insert punctuation mark X before/after Y”, but none is of the form “insert punctuation mark X between Y and Z”, we decided to build two different models for “preceding punctuation” and “following punctuation”. At runtime, at each juncture between two words, both models are queried for each non-terminal node in the parent chain. If one of them indicates a high probability of a certain punctuation mark, that vote wins out and the punctuation mark is inserted.

Input features

The features for the punctuation models are different from the feature sets used in other models. Most of the features are special features, checking the tree configuration:

- Nodetype and Nodetype of the head

- Nodetype of the parent and Nodetype of the head of the parent

- ParentAttrs on the SemNode

- SentenceLengthInToken and SentenceLengthInChar
- AtRightEdgeOfParent/AtLeftEdgeOfParent: indicating whether the node is at the right/left edge of its parent node
- NumTokens and NumChars: number of tokens/chars of the node
- DistanceToSentenceInitialInToken and DistanceToSentenceFinalInToken
- DistanceToSentenceInitialInChar and DistanceToSentenceFinalInChar
- FirstLemma and LastLemma
- NodetypeOfLeftMostDaughter and NodetypeOfRightMostDaughter
- NodetypeOfTopRightEdge and NodetypeOfTopLeftEdge: Nodetype of the largest ancestor node with the same right/left edge
- NodetypeOfLargestPreceedingNT and NodetypeOfSmallestPreceedingNT: Nodetype of the largest/smallest preceding non-terminal node
- NodetypeOfLargestFollowingNT and NodetypeOfSmallestFollowingNT: Nodetype of the largest/smallest following non-terminal node

Features selected

In the model for preceding punctuation, all of the features listed above were selected, with the exception of NodetypeOfTopRightEdge. In the model for following punctuation, two features were not selected: DistanceToSentenceFinalInToken and SentenceLength-InToken.

Classifier accuracy and complexity

The accuracy of the model for preceding punctuation is 98.65%, with a baseline of 89.61%.

Key	Precision	Recall	F-measure
COMMA	0.9500(29727/ 31293)	0.9228(29727/ 32213)	0.9362
OTHERS	0.0000(0/ 0)	0.0000(0/ 23)	0.0000
DASH	0.0000(0/ 0)	0.0000(0/ 83)	0.0000
SEMICOLON	0.0000(0/ 0)	0.0000(0/ 33)	0.0000
NULL	0.9907(280067/ 282705)	0.9945(280067/ 281610)	0.9926
COLON	0.8153(203/ 249)	0.7123(203/ 285)	0.7603

Figure 37: Precision, recall, and F-measure for the preceding punctuation model

The accuracy of the model for following punctuation is 98.48% with a baseline of 94.98%.

Key	Precision	Recall	F-measure
COMMA	0.8795(12462/ 14169)	0.8084(12462/ 15416)	0.8425
DASH	0.0000(0/ 0)	0.0000(0/ 45)	0.0000
SEMICOLON	0.0000(0/ 0)	0.0000(0/ 27)	0.0000
NULL	0.9897(294194/ 297241)	0.9943(294194/ 295884)	0.9920
COLON	1.0000(125/ 125)	0.7669(125/ 163)	0.8681

Figure 38: Precision, recall and F-measure for the following punctuation model

Failure analysis

Figure 37 and Figure 38 show that predictions are reliable for commas, and somewhat reliable for colons. For other punctuation, such as dash and semicolon, data are simply too sparse.

7.2 The Order Model

7.2.1 Motivation

Word order plays a crucial role in establishing the fluency and the intelligibility of a sentence. As section 3 explains, word order can make the difference between sensibility and gibberish, especially in a German sentence. Given a syntax tree for a sentence with unordered constituents, such as the tree in Figure 39, the goal of the Amalgam ordering stage is to establish linear order within each constituent, so that the head and each modifier are placed in their proper position. The ordering stage handles each constituent independently and in isolation, but the net effect is to establish linear order among all leaves of the tree. In our example, the constituent DECL3 has head PREFIX1 (head denoted in the figure by the asterisk '*') with modifiers VP2, NP5, NP6, and RELCL3. The ordering stage places these in order independently of the head and modifiers of other constituents, such as RELCL3, for example.

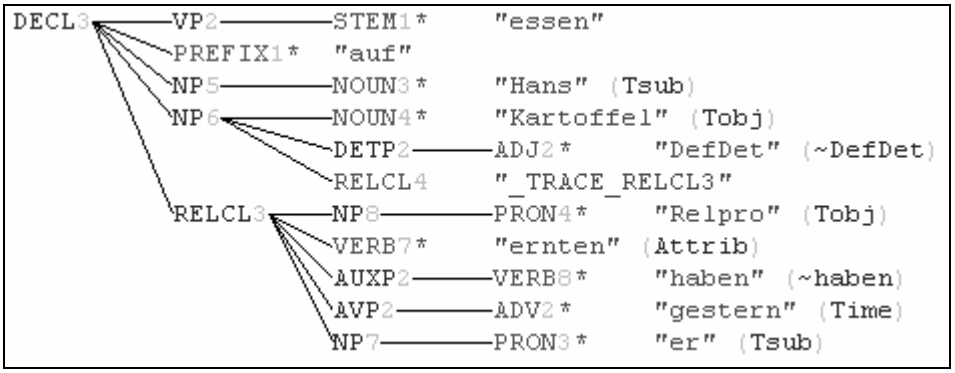


Figure 39: The syntax tree for the sentence *Hans isst die Kartoffeln auf, die er gestern geernet hat* before ordering

Figure 40 displays the tree after being ordered. The ordering stage has placed the children of DECL3 in the order NP5, VP2, NP6, PREFIX1, and RECLCL3.

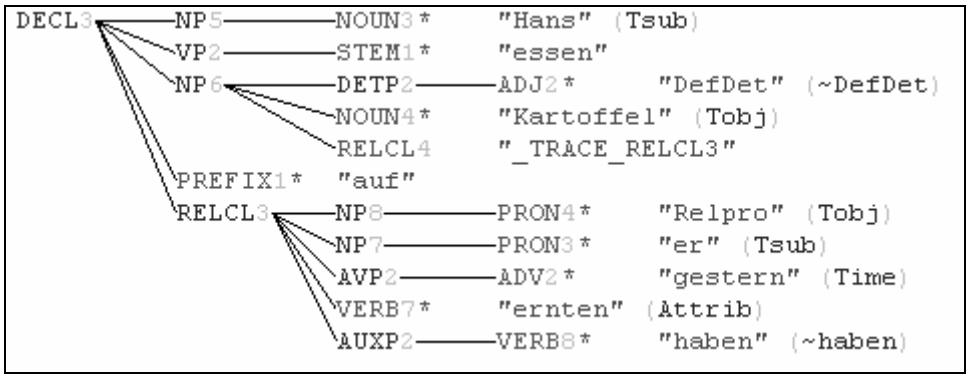


Figure 40: The syntax tree after ordering

7.2.2 Model and Features

The Amalgam ordering stage employs a generative statistical model of syntactic tree structure to score possible orders among a head and its modifiers. The term "generative" refers to the fact that the distributions in the model could be sampled to generate or build actual syntax trees from scratch, in distribution consistent with the model. It is a useful conceptual framework, even though we are not actually creating the tree at this point in the sentence realization process.

For a given constituent, the model assigns a probability to modifier sequences in the context of several relevant features. Many features can be used as relevant context; in practice, our implemented model currently employs the following features:

- nodetype of the parent of the head (i.e., the constituent type),
- nodetype of the head (i.e., head part-of-speech)

Other possible contextual features include:

- lemma of the head
- verb position bits on the parent of the head
- nodetype of the grandparent of the head
- presence of an auxiliary in the constituent

Given the context features, the model assigns probability to a modifier sequence. Currently each modifier consists of two features:

- semantic relation (from the logical form) of the modifier to the head
- nodetype (part-of-speech) of the modifier

Other possible features of a modifier include:

- lemma of the modifier

The model is currently constructed to approximate modifier sequence probabilities with an n-gram model. Given a particular context, the model assigns a probability to the semantic relation (from logical form) of each modifier, in the constituent's context and in the context of the preceding n-1 neighbors, and it assigns a probability to the nodetype (syntactic category) of the modifier. In the current system, the number of preceding neighbors currently considered is only one; hence, the order model employs a context-dependent bigram. Here is a schematic for a constituent that illustrates the context of parent and head as well as the pre-modifiers and post-modifiers:

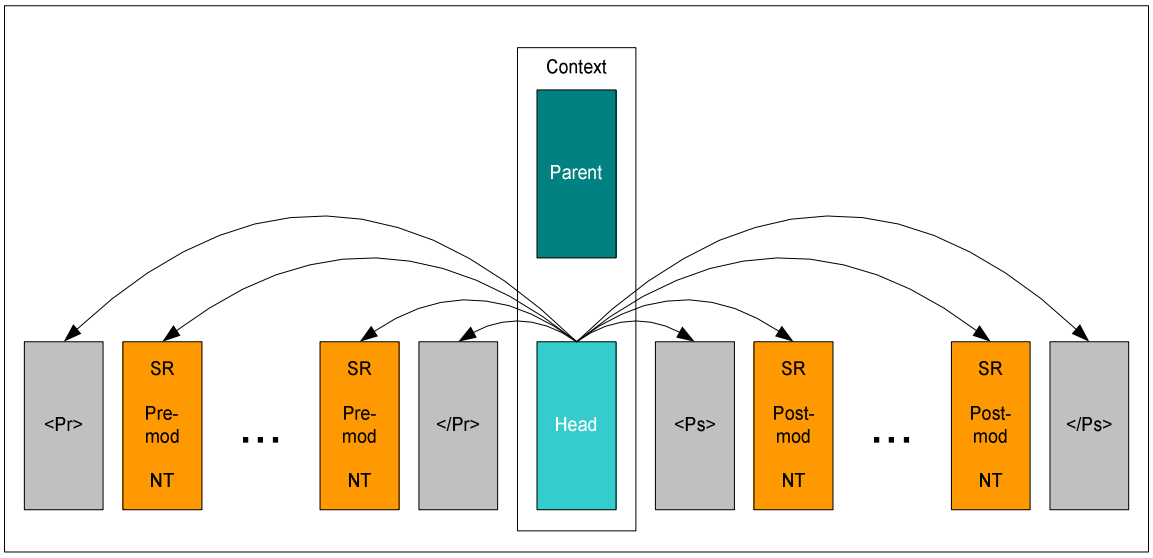


Figure 41: Constituent order schematic

The model is split into a model of head pre-modifier order (on the left of Figure 41) and of head post-modifier order (on the right of the figure). Included in the notion of modifier are explicit pseudo-modifiers for marking the beginning and end of the pre-modifiers (<Pr> and </Pr>, respectively) and for marking the endpoints of the post-modifiers (<Ps> and </Ps>), as shown in the figure. Hence, for any Parent/Head context, the model includes an n-gram distribution for pre-modifiers and an n-gram distribution for post-modifiers. All such distributions are encoded in a single model file. Figure 42 contains a fragment of a model file for illustrative purposes:

```
[356/563] ( DECL VERB ) </Pr> </Pr> Time : AVP
[16/563] ( DECL VERB ) </Pr> </Pr> Time : AVPNP
[10/563] ( DECL VERB ) </Pr> </Pr> Time : NP
[168/563] ( DECL VERB ) </Pr> </Pr> Time : PP
[13/563] ( DECL VERB ) </Pr> </Pr> Time : SUBCL
```

Figure 42: Model file fragment for the nodetype feature of a pre-modifier with semantic relation Time

It shows the context with a DECL (declarative sentence node) as parent and a VERB as head. The fragment shows the distribution for the nodetype feature of a pre-modifier with semantic relation "Time" preceding the "</Pr>" marker (working out from the head). As

indicated, such a modifier has probability 356/563 of being an AVP, 16/563 for AVPNP, 10/563 for NP, 168/563 for PP, and 13/563 for SUBCL in that context.

7.2.3 Search and Complexity

The ordering stage must search among all possible orders or at least among the most promising orders. The search proceeds by considering all possible incomplete orderings of length one, then two, and so on, up to all possible complete orderings of length n. Each step in the search can be pruned to consider only those incomplete order hypotheses for which the model assigns a sufficiently high score. This search is capable of producing as many scored order hypotheses as one cares to retrieve from the final step of the search and is commonly called a "beam search," since the threshold for determining a "sufficiently high score" is often termed the "beam."

For n members (counting the head and its modifiers), there are n! possible orderings; hence the search space can be overwhelmingly large for a heavy constituent. The beam search constrains the complexity of the complete search and is nearly optimal.

8 Performance

Performance of Amalgam was evaluated on a 550MHz PC. On a set of 260 sentences from the technical domain (computer manuals), generation time from logical forms (without the analysis part) was 0.30 seconds per sentence or 3.25 sentences per second. The sentences in the test file had an average length of 15 words per sentence.

9 Evaluation

In April 2002, we evaluated the overall system by parsing a blind and randomized test set of 564 German sentences[7] to produce logical forms and then applying Amalgam to generate output strings from those logical forms. For this sample, 71.1% of the words are correctly inflected and occur in the correct position in the sentence. We also compute the word-level string edit distance of the generated output from the original reference string: the number of errors (insertions, deletions, and substitutions) is 44.7% of the number of words in the reference string.

String edit distance is a harsh measure of sentence realization accuracy. Because string edit distance does not consider movement as an edit operator, movements appear as both deletions and insertions, yielding a double penalty. Furthermore, as observed earlier, some edits have a greater impact on the intelligibility of the output than others, especially the position of the German verb. Work in progress on a tree edit distance metric addresses both of these issues (Ringger et al., in preparation). Closely related work includes that of Bangalore, Rambow and Whittaker (2000).

It is possible that generated sentences might differ from the reference sentences and yet still prove satisfactory. We therefore had five independent human evaluators assess the

[7] We extract a random sample of generated sentences. We take the first *n* sentences in the sample necessary to ensure 500 sentences that differ from the reference sentence.

quality of the output for that same blind and randomized test set of 564 sentences. These sentences had been analyzed to yield logical forms from which Amalgam generated output sentences. The evaluators assigned an integer score to each sentence, comparing it to the reference sentence using the scoring system given in Table 1.[8]

The average score was 2.96 with a standard deviation of 0.81. The mode was 4, occurring 104 times, i.e. 104/564 sentences, or 18.4% received the maximum score. In 63 of these cases, the score of 4 had been automatically assigned because the output sentence was identical to the reference sentence. In the other 41 cases, all five human evaluators had assigned a score of 4, i.e., the output differed from the reference sentence, but was still "Ideal".

1. "Unacceptable". Absolutely not comprehensible and/or little or no information transferred accurately.
2. "Possibly Acceptable". Possibly comprehensible (given enough context and/or time to work it out); some information transferred accurately.
3. "Acceptable". Not perfect (stylistically or grammatically odd), but definitely comprehensible, AND with accurate transfer of all important information.
4. "Ideal". Not necessarily a perfect translation, but grammatically correct, and with all information accurately transferred.

Table 1: Evaluation guidelines

10 Using Amalgam in Machine Translation: First Results

After we had implemented the German-to-German Amalgam prototype, we started using it in a machine translation context. In machine translation, the logical form representation that serves as input to Amalgam is not produced from the analysis of a German sentence, but is transferred through learned mappings between source language logical forms and German logical forms. For our first machine translation experiments the source language was English. For details of the mapping process and the setup of the machine translation system see (Richardson et al. 2001a) and (Richardson et al. 2001b).

A first adaptation that was necessary to make Amalgam work properly in this context was to reduce the features used in learning the models to those that are actually available on transferred logical forms. The result of retraining our Amalgam models on this smaller set of features was very encouraging: none of the models exhibited any significant drop in accuracy.

The challenge of using Amalgam in machine translation is that Amalgam is trained on "native" German logical forms. To the extent that transferred logical forms correspond closely to native target language logical forms, the results are close to what we saw in German-to-German generation. Problems arise, however, when the transferred logical forms exhibit properties that are not found in native logical forms. Ideally, of course, the transfer component of the machine translation system should produce perfectly native-

[8] These guidelines were originally intended for assessing the quality of machine translation, measuring fluency and transfer of semantic content from the source language. Evaluating the sentence realization component is conceptually a case of German-to-German translation.

like logical forms, but this is an area of ongoing research - especially in a multi-lingual machine translation setup where the transfer component should not be fine-tuned to a specific language-pair, but should be broad and general enough to accommodate languages as different as Chinese, Japanese, German, English, French and Spanish. We are currently researching the possibilities of learning filters that post-process the transferred logical forms and adjust certain feature values to what we would expect in a native German logical form before those logical forms are input to the Amalgam module.

First results on the quality of Amalgam output in machine translation compared to state-of-the-art commercially available English-to-German machine translation systems are encouraging. In April 2002, an independent agency (the Butler Hill Group) conducted a first baseline evaluation of our English-to-German translation system compared to the Saillabs system. Six evaluators compared the output of both systems (in randomized order and with anonymized source) to a reference translation. Two hundred fifty sentences from the technical domain were evaluated. The raters ranked each sentence according to a three-way distinction: 1 if the Nlpwin system was better, -1 if the comparison system was better, and 0 if both systems were equally good or bad. The result of this evaluation was that the output of both systems, while relatively poor, is rated equally: the average score was -0.069 with a +/-0.11 confidence interval at a 0.89 significance level.

After implementing a prototype of the filter discussed above, adding a compound generation function and after some low-level bug-fixes, we conducted a second evaluation one month later. At this time, the average score was 0.13 with a confidence interval of +/- 0.12 at the 0.99 significance level. The number of sentences that were rated as being better in the Nlpwin system jumped from 103 in April to 131 in the second evaluation.

11 Conclusion

We have described the current state of our ongoing research into sentence realization, blending machine-learned and knowledge-engineered approaches.

We are currently working with colleagues to implement Amalgam for French sentence realization. This will serve as a useful test of the extent to which the Amalgam architecture is language-independent.

We continue to refine the decision tree classifiers and the ordering model. We are also experimenting with machine-learned approaches to resolving underspecified or noisy logical forms that Amalgam encounters in the context of machine translation.

Finally we intend to experiment with a beam search throughout the sentence realization process, propagating the top hypotheses from each decision tree classifier rather than applying a greedy search as is presently the case.

12 Acknowledgements

Our thanks go to Max Chickering for his very generous help with the WinMine toolkit. Zhu Zhang made significant contributions to the modeling of punctuation and extraposition as an intern during the summer of 2001. Tom Reutter of the NLP group implemented the inflectional generation component for German during the German grammar checker project, and he continued to improve inflectional generation based on feedback from error analysis in Amalgam. The Butler Hill Group, and especially Karin Berghöfer, have assisted in evaluation and error analysis. Last, but not least, the input from our colleagues in the NLP group has contributed much to the progress of this project.

References

Aikawa, T., M. Melero, L. Schwartz and A. Wu. 2001a. Multilingual sentence generation. *Proceedings of the 8th European Workshop on Natural Language Generation*. Toulouse, France. 57-63.

Aikawa, T., Melero, M., Schwartz, L. and Wu, A. 2001b. Generation for Multilingual MT. *Proceedings of the MT-Summit*, Santiago de Compostela, Spain. 9-14.

Bangalore, S. and O. Rambow. 2000a. Exploiting a probabilistic hierarchical model for generation. *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*. Saarbrücken, Germany. 42-48.

Bangalore, S. and O. Rambow. 2000b. Corpus-based lexical choice in natural language generation. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*. Hong Kong, PRC. 464-471.

Bangalore, S., O. Rambow, and S. Whittaker. 2000. Evaluation metrics for generation. *International Conference on Natural Language Generation* (INLG 2000). Mitzpe Ramon, Israel. 1-13.

Bontcheva, K. and Y. Wilks. 2001. Dealing with dependencies between content planning and surface realization in a pipeline generation architecture. *Proceedings of IJCAI 2001*. 1235-1240.

Chickering, David Maxwell. nd. WinMine Toolkit Home Page. http://research.microsoft.com/~dmax/WinMine/Tooldoc.htm

Corston-Oliver, Simon. 2000. Using Decision Trees to Select the Grammatical Relation of a Noun Phrase. *Proceedings of the 1st SIGDial workshop on discourse and dialogue*. Hong Kong, PRC. 66-73.

Corston-Oliver, S., M. Gamon, E. Ringger, R. Moore. 2002. An overview of Amalgam: A machine-learned generation module. To appear in *Proceedings of the International Natural Language Generation Conference*. New York, USA.

Dalianis, H. and E. Hovy. 1993. Aggregation in natural language generation. Giovanni Adorni and Michael Zock (eds), *Trends in Natural Language Generation—An Artificial Intelligence Perspective*. 88-105.

Duboue, P. and K. McKeown. 2001. Empirically estimating order constraints for content planning in generation. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (ACL-2001). Toulouse, France. 172-179.

Eisenberg, P. 1999. *Grundriss der deutschen Grammatik. Band2: Der Satz*. Metzler, Stuttgart/Weimar.

Elhadad, M. 1992. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. PhD Thesis, Columbia University.

Engel, U. 1996. *Deutsche Grammatik.* Groos, Heidelberg.

Gamon, M., E. Ringger, S. Corston-Oliver, R. Moore. 2002a. Machine-learned contexts for linguistic operations in German sentence realization. To appear in *Proceedings of The Fortieth Anniversary Meeting of the Association for Computational Linguistics*. Pennsylvania, PA, USA.

Gamon, M., E. Ringger, Z. Zhang, R. Moore, S. Corston-Oliver. 2002b. Extraposition: A case study in German sentence realization. To appear in *Proceedings of the 19th International Conference in Computational Linguistics (COLING) 2002*. Taipei, Taiwan..

Goldsmith, J. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics* 27:153-198.

Halliday, M.A.K. 1985. *An Introduction to Functional Grammar*. Edward Arnold, London.

Hitzeman, J., C. Mellish and J. Oberlander. 1997. Dynamic generation of museum web pages: The intelligent labelling explorer. *Journal of Archives and Museum Informatics* 11:107-115.

Kay, M. 1979. Functional grammar. *Proceedings of the Fifth Meeting of the Berkeley Linguistics Society,* Berkeley, California, USA. 142-158.

Jensen, K., G. E. Heidorn and S. Richardson. 1993. *Natural Language Processing: The PLNLP Approach*. Kluwer, Boston/Dordrecht/London.

Joshi, A. 1987. The relevance of tree adjoining grammars to generation. In G. Kempen (ed.) *Natural Language Generation: New Directions in Artificial Intelligence, Psychology and Linguistics.* Kluwer, Dordrecht.

Knight, K., I. Chander, M. Haines, V. Hatzivassiloglou, E. Hovy, M. Ida, S.K. Luk, R. Whitney and K. Yamada. 1995. Filling knowledge gaps in a broad-coverage MT system. *Proceedings of the 14th IJCAI Conference*, Montréal, Québec, Canada. 1390-1397

Langkilde, I. nd. Thesis proposal: Automatic sentence generation using a hybrid statistical model of lexical collocations and syntactic relations. Ms.

Langkilde, I. and K. Knight. 1998a. The practical value of n-grams in generation. *Proceedings of the 9th International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada. 248-255.

Langkilde, I. and K. Knight. 1998b. Generation that exploits corpus-based statistical knowledge. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998)*. Montréal, Québec, Canada. 704-710.

Malouf, R. 2000. The order of pronominal adjectives in natural language generation. *Proceedings of 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, PRC. 85-92.

Mann, W. and S. Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text* 8(3). 243-281.

McCawley, J. D. 1988. *The Syntactic Phenomena of English*. The University of Chicago Press, Chicago and London.

Mel'čuk, I. 1988. *Dependency Syntax: Theory and Practice.* State University of New York Press, Albany, NY.

Mellish, C., A. Knott, J. Oberlander and M. O'Donnell. 1998. Experiments using stochastic search for text planning. *Proceedings of the International Conference on Natural Language Generation*. 97-108.

Meteer, M. 1989. The SPOKESMAN natural language generation system. Report 7090, BBN Systems and Technologies, Cambridge, Massachusetts, USA.

Oberlander, Jon and Chris Brew. 2000. Stochastic text generation. To appear in *Philosophical Transactions of the Royal Society of London, Series A*, volume 358.

Penman. 1989. The Penman documentation. Technical Report. USC/ISI..

Poesio, M., R. Henschel, J. Hitzeman and R. Kibble. 1999. Statistical NP generation: A first report. *Proceedings of the ESSLLI Workshop on NP Generation*. Utrecht, Netherlands.

Ratnaparkhi, A. 2000. Trainable methods for surface natural language generation. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics (ANLP-NAACL 2000)*. Seattle, Washington, USA. 194-201.

Reape, M. and Mellish C. 1999. Just what is aggregation anyway? *Proceedings of the 7th European Workshop on Natural Language Generation*. Toulouse, France.

Reiter, E. and C. Mellish. 1992. Using classification to generate text. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-1992).* University of Delaware, Newark. 265-272.

Reiter, E., C. Mellish and J. Levine. 1992. Automatic generation of on-line documentation in the IDAS project. *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP-1993)*, Trento, Italy. 64-71.

Reiter, E. 1994. Has a consensus NL generation architecture appeared, and is it psychologically plausible? *Proceedings of the 7th International Workshop on Natural Language Generation.* Kennebunkport, Maine. 163-170.

Richardson, S., W. Dolan and L. Vanderwende. 1998. MindNet: Acquiring and structuring semantic information from text. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998).* Montréal, Québec, Canada. 1098-1102.

Richardson, S., W. Dolan, A. Menezes, and M. Corston-Oliver. 2001a. Overcoming the customization bottleneck using example-based MT. *Proceedings, Workshop on Data-driven Machine Translation, 39th Annual Meeting and 10th Conference of the European Chapter, Association for Computational Linguistics*. Toulouse, France. 9-16.

Richardson, S., W. Dolan, A. Menezes, and J. Pinkham. 2001b. Achieving commercial-quality translation with example-based methods. *Proceedings of the VIIIth MT Summit*. Santiago de Compostela, Spain. 293-298.

Ringger, E., R. Moore, M. Gamon, S. Corston-Oliver. In preparation. A tree edit distance metric for evaluation of natural language generation.

Scott, D., R. Power and R. Evans. 1998. Generation as a solution to its own problem. *Proceedings of the European Conference on Artificial Intelligence*. 677-681.

Shaw, J. 1998 Segregatory Coordination and Ellipsis in Text Generation. *Proceedings of COLING-ACL*. 1220-1226.

Stolcke, A. 1997. Linguistic knowledge and empirical methods in speech recognition. *AI Magazine* 18(4):25-31.

Uszkoreit, H., T. Brants, D. Duchier, B. Krenn, L. Konieczny, S. Oepen and W. Skut. 1998. Aspekte der Relativsatzextraposition im Deutschen. Claus-Report Nr.99, Sonderforschungsbereich 378, Universität des Saarlandes, Saarbrücken, Germany.

Walker, M., O. Rambow, and M. Rogati. 2001. SPoT: A trainable sentence planner. *Proceedings of the North American Meeting of the Association for Computational Linguistics.*

Yamada, K. And K. Knight. 2001. A syntax-based statistical translation model. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguisitics* (ACL-2001). Toulouse, France. 523-529.

Zukerman, I., R. McConachy and K. Korb. 1998. Bayesian reasoning in an abductive mechanism for argument generation and analysis. *AAAI98 Proceedings -- the Fifteenth National Conference on Artificial Intelligence.* 833-838, Madison, Wisconsin.

Wilkinson, J. 1995. Aggregation in Natural Language Generation: Another Look. Co-op work term report, Department of Computer Science, University of Waterloo.