

Using an Annotated Corpus As a Knowledge Source For Language Generation

Tomasz Marciniak and Michael Strube

EML Research gGmbH

Schloss-Wolfsbrunnenweg 33

69118 Heidelberg, Germany

<http://www.eml-research.de/nlp>

Abstract

We present an approach to building a realization component for an NLG system that makes use of an annotated corpus as the source of linguistic knowledge. Three issues that we focus on include: modeling of the generation decisions, corpus annotation and specification of the system architecture.

1 Introduction

Linguistic realization, the *tactical* part of Natural Language Generation (NLG), can be defined as a mapping between the conceptual content of a linguistic expression and its grammatical form. To perform this mapping, an NLG system requires a substantial amount of knowledge of how lexical and syntactic constructions can be applied at different levels of the linguistic organization (i.e. *clause, discourse*) to encode the intended meaning.

In this paper we present an approach to building a trainable realization component applicable to narrow domains such as *route directions*. Instead of relying on hand-crafted linguistic resources, such as a grammar or a lexicon, the system acquires all the lexical and syntactic knowledge necessary for linguistic realization from a small, domain-specific corpus. Hence, the task of collecting a corpus and annotating it with the required semantic and grammatical information becomes an integral part of system development.

The methodology we propose assumes three stages: modeling of the generation decisions, preparation of the corpus and specification of the processing flow. The first step involves a semantic and grammatical analysis of the target texts, and its goal is to identify low-level decisions that would drive the generation process. A corpus of relevant target texts must be then collected and annotated with the required semantic and grammatical information. The final step is to determine the architecture that would specify the flow of information between modules handling individual decisions. We compare two such architectures: a sequential model which assumes a strict ordering of generation decisions and an integrated model, based on an Integer Linear Programming formulation.

The paper is structured as follows: in Section 2 we discuss the use of corpora in NLG. Sections 3 and 4 are concerned respectively with modeling the semantics and grammatical

form of route directions. In Section 5 we describe the annotation process and in Section 6 we introduce two implemented architectures and report on the evaluation results.

2 The Use of Corpora in NLG

The main function of a corpus in generation is to provide a basis for *requirements analysis* [Reiter and Dale, 2000]. It aims at determining the target texts, characterizing their structure and identifying domain-specific linguistic constructions used to communicate the input data. In the traditional *rule-based* approach to generation, the information gathered in this way can be used by system developers to explicitly model the behavior of the system.

Alternatively, a corpus can be treated as a source of data from which the generation knowledge can be extracted using statistical and machine learning techniques. This approach can be used to construct trainable components which do not require extensive qualitative analysis and domain expertise on the part of the developer and help to overcome the knowledge acquisition bottleneck. Using this approach, the developer cannot influence directly the choices that the system makes. It is the quality of the corpus and the right formulation of the generation decisions that determine the quality of the output texts (cf. [Reiter *et al.*, 2003]). The advantage of using corpus-based techniques, is that they can be ported to new domains and applications. They also help to bridge the gap between aspects of generation traditionally viewed as separate, such as sentence vs. discourse generation, or lexicalization vs. syntactic realization.

2.1 Related Work

Corpus-based methods were introduced to NLG in the context of syntactic realization [Langkilde and Knight, 1998]. Most current works in this area follow the *ranking* approach which involves rule-based overgeneration and then corpus-driven selection of the best candidate, e.g. [Bangalore and Rambow, 2000b; Varges and Mellish, 2001]. Empirical techniques were also applied to solving isolated tasks, such as lexical choice, e.g. [Bangalore and Rambow, 2000a], ordering of NP modifiers, e.g. [Shaw and Hatzivassiloglou, 1999] or sentence ordering, e.g. [Lapata, 2003]. In these works, corpus-based methods were applied to solving individual tasks in a non-application context. To the best of our knowledge, a few

projects only tried to cover the whole process of linguistic realizations relying exclusively on trainable components [Chen *et al.*, 2002; Kan and McKeown, 2002].

2.2 Our Goals

In the current work we investigate the use of a single corpus as the sole source of linguistic knowledge necessary to drive the process of linguistic realization. In formal terms, we can define this process as a one-to-many mapping between a representation of the semantic content of an expression and its linguistic form. We concentrate on three important issues that development of a corpus-based realization component involves:

- abstract modularization of the generation process in terms of discrete, low-level tasks whose realization is to be learned from the corpus,
- annotation of the corpus with the semantic and grammatical information, corresponding respectively to the input and output of the system, and
- specification of the system architecture which determines the processing flow.

We illustrate this three-step development process on the example of *route directions* generation.

3 Route Directions

To identify the target texts in our domain we first performed an informal analysis of human-authored route directions available on the Internet. We identified three categories of such texts, characterized by different levels of the linguistic and conceptual complexity:

1. Directions consisting of phrasal expressions only, with no verbs or discourse connectives, e.g. *Down the street, past the post office, left onto Church Street.*
2. Directions expressed with imperative clauses only, with a limited number of verbs and discourse connectives, e.g. *Leave the building, turn right onto Dowman, go to 24th Street.*
3. Directions expressed with complex texts, characterized by a broad lexical (verbs, connectives) and structural variation, (see Example 1 below).

Example 1. (a) Standing in front of the hotel (b) follow Meridian street south for about 100 meters, (c) passing the First Union Bank entrance on your right, (d) until you see the river side in front of you. (e) Then make a left onto North Hills Street. (f) The auditorium will be up the street on your left.

We decided to collect a corpus of texts from category (3) for two reasons¹. Firstly, we wanted to ensure that the generator can express complex content in a consistent manner and hence decided not to mix different types of texts (cf. [Reiter and Sripada, 2002]). Secondly, we noticed structural similarities between route directions from (3) and other types of

¹Notice, however, that there is no clear-cut boundary between categories (2) and (3).

instructional texts, such as cooking or assembly instructions, which offered a promise of reusing, at least partially, the annotated corpus.

In our application, the task of generating route directions comprises two stages. The *strategic* part is concerned with mapping a topological specification of a route, i.e. a *route plan* onto a dynamic model organized around a set of temporally related *situations*. In the domain of route directions, situations schematize the spatio-temporal interactions between the route follower and salient elements of the environment (e.g. streets, landmarks). The *tactical* part of the generation process, on which we focus in this paper, consists in constructing the linguistic description of the underlying content.

3.1 Ontology of Situations

The ontology of situations constitutes a semi-formal specification of the conceptual content of route directions, and as such sanctions the input to the realization component. It spans three conceptual levels, each having its own taxonomy of classes and properties, related to one another through axioms specifying well-formedness conditions (see [Marciniak and Strube, 2005b] for a detailed discussion).

At the *aspectual level*, a situation is assigned to a specific aspectual category such as: *state*, *process*, *accomplishment* or *achievement* [Vendler, 1967]. The membership in a given category is further formalized as a function of three binary attributes: *stative*, *durative* and *culminated* [Moens and Steedman, 1988].

At the *frame level*, the conceptual category of each situation is specified. In our domain, three main categories are *SelfMotion*, *Localization* and *VisualPerception*. Each of them is modeled as a frame, with slots specifying conceptual roles attributed to situation participants (see Table 1)².

Frame	Conceptual Roles
SELF MOTION	self_mover, source, path, goal, etc.
VISUAL PERCEPTION	perc_object, location, direction, etc.
LOCALIZATION	loc_object, location, direction, etc.

Table 1: Frames with corresponding conceptual roles

Finally, situations are not isolated, but occur in temporally structured groups. Three temporal relations that we consider are: *initialRelation*, *ongoingRelation* and *subsequentRelation*. The relations are functional and non-reversible, which results in the structure taking the form of a tree (see Figure 1). For each pair of related situations, the *child node* situation is recognized as the *trajector* being located relative to the *landmark* situation (i.e. during its *initial*, *ongoing* or *subsequent* stage). This trajector-landmark asymmetry finds direct manifestation in the corresponding linguistic form. In Example 1, for instance, situation denoted by (d) bears the *linguistic marking* (i.e. discourse connective), which signals its relation to the situation from (b)³. Hence, to properly realize a linguistic description of a situation, its temporal context must be considered.

²The categories are partially modeled after FrameNet *frames* and roles correspond to *frame elements*, cf. [Baker *et al.*, 1998])

³Temporal relations can be also signaled by the verb form e.g. *gerund* in (a,c), or simply the linear ordering, e.g. (d) < (e)

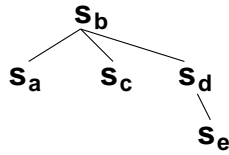


Figure 1: Temporal structure of situations. Nodes s_a , s_b , etc. correspond to discourse units (a), (b), etc. from Example 1.

4 Modeling Generation Decisions

Linguistic realization of route directions comprises a range of different types of generation decisions. At the clause level, verbs and prepositions must be lexicalized, and the syntactic frame of the clause must be specified⁴. At the discourse level, clauses must be ordered, and where appropriate, discourse connectives lexicalized. To be able to learn these decisions from a corpus, we first need to determine their scope⁵, and then formulate them as classification problems.

In the approach presented here, we associate low-level generation decisions with *minimal* elements of the grammatical form, which may affect the meaning of an expression or render it ill-formed.

4.1 LTAG-based Representation

To determine the range of generation decisions relevant to our application, we first model the grammatical form of route directions using Lexicalized Tree Adjoining Grammar (LTAG) [Joshi and Schabes, 1991]. LTAG provides a useful abstraction of the process of building the grammatical form of an expression, called *derivation*. It starts with the selection of *elementary trees*, anchored by lexical items, such as verbs or prepositions. In the next step, elementary trees are assembled by means of well-defined operations of *substitution* and *adjunction*. Originally used to model the sentence derivation only, LTAG has been shown to apply equally well to the structure of a discourse (cf. [Webber and Joshi, 1998]). At the discourse level, each clause in a text is associated with an elementary tree anchored by a discourse connective. Following the underlying temporal or rhetorical relations, discourse-level trees are combined to form the derived structure.

As an example, consider the derivation of discourse unit (c) from Example 1 in the context of (a), (b) and (d) (Figure 2). At the clause level, the set of elementary trees includes one *initial* tree α_1 anchored by the main verb which projects to S and specifies the syntactic frame of the clause, and *auxiliary* trees β_1 and β_2 corresponding to the verb arguments. The auxiliary trees are successively adjoined to VP node of α_1 immediately dominated by S . The order in which the adjunctions take place determines the final ordering of the arguments in the clause.

At the discourse level (cf. Figure 3), the discourse unit describing the root situation from Figure 1 is modeled as the initial tree α_2 , and auxiliary trees β_3 and β_4 represent the related discourse units. Following the temporal dependencies,

⁴In our application, nominal expressions are own names mostly, e.g. *Meridian Street*, and as such are directly specified in the input, hence we ignore here the problem of NP generation.

⁵E.g. decide whether specifications of the lexical and grammatical form of a verb constitute a single decision, i.e. *passing* or if they are determined separately: *pass* + *gerund*

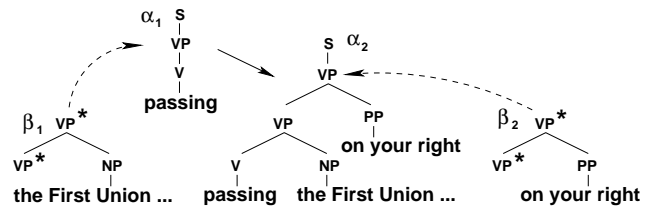


Figure 2: Clause-level derivation

trees β_3 and β_4 are adjoined to α_2 . Again, the ordering of the operations determines the linear structure of the text.

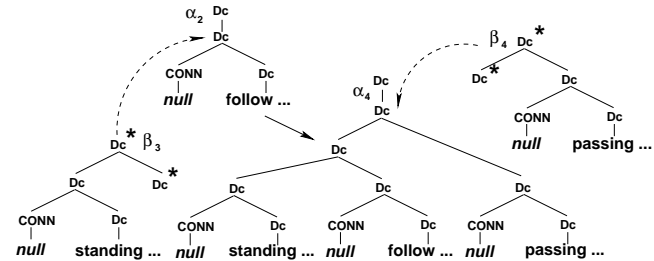


Figure 3: Discourse-level derivation

4.2 Feature Vector Encoding

To model the whole process in a uniform way we represent elementary trees with feature-value pairs necessary to account for the distinctions between individual trees and determine the order of adjunction operations (see Table 2).

Elementary Tree	Feature	Possible Values
Main Verb	s_exp	$NP, NP-VP$
	$verb_lex$	$walk, follow, pass, etc.$
	$verb_form$	$bare_inf, gerund, etc.$
Verb Argument	adj_rank	$numeric$
	phr_type	NP, PP, P
	$prep_lex$	$to, past, towards, etc.$
Discourse Unit	adj_rank	$numeric$
	adj_dir	$left, right$
	$connective$	$null, and, until, etc.$

Table 2: Frames with corresponding conceptual roles

The tree associated with the main verb in a clause is represented with three features: s_exp , $verb_lex$ and $verb_form$, denoting respectively whether a clause should have an explicit subject (i.e. $NP-VP$), the lexical form of the verb and its grammatical form. For each verb argument, the corresponding tree is modeled with features: adj_rank which determines the ordering of the adjunction operations, phr_type which specifies the syntactic category of the argument, and $prep_lex$, i.e. the lexical form of the preposition/particle (if phr_type is other than NP). Finally, the discourse-level trees representing discourse units are modeled with three features: adj_rank and adj_dir denoting respectively the ordering of the adjunction operations and the adjunction direction and $connective$ specifying the lexical form of the discourse connective (or $null$ if no explicit connective is present).

4.3 Generation Decisions Revisited

Realizations of individual features used to encode the LTAG derivation are now taken to constitute single generation deci-

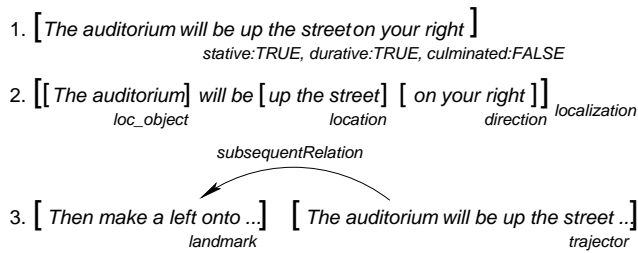


Figure 4: Three levels of annotation: 1. *Aspectual Level*, 2. *Frame Level* and 3. *Temporal Level*

sions. Each such decision constitutes a classification problem and consists in assigning a class label to an instance specifying the semantic context of the particular decision.

Notice that this formulation draws no distinctions between lexical and syntactic decisions on the one hand, and discourse-level and clause-level decisions on the other. This fact simplifies the procedure of acquiring the necessary knowledge from the corpus. For each decision, a classifier handling a single-valued prediction needs to be *induced* from a set of training instances.

5 Corpus Annotation

To obtain the training data, we collected a corpus of 75 route directions texts, with a total number of 904 discourse units. By applying the concept of *stand-off* annotation, i.e. separation of tags from the actual texts, we were able to realize the annotations at multiple levels. Each annotation level comprises a set of *markables*, i.e. marked text spans associated with the required linguistic information and falling in four categories:

- *Discourse Unit* markables, corresponding to individual clauses,
- *Predicate* markables, corresponding to main verbs within clauses,
- *Argument* markables, corresponding to phrasal arguments of verbs, and
- *Connective* markables, corresponding to conjunctions and discourse markers.

During the pre-processing stage the texts were tokenized, POS-tagged, and markables were automatically detected with a simple, rule-based system, tuned to the given type of texts.

5.1 Semantic Annotation

In order to apply the ontological model of situations to the annotation task we defined an annotation scheme comprising a selection of semantic tags which provide a flat representation of the categories specified in the ontology (Table 3).

Annotations have been realized at different levels, corresponding to the levels specified in the ontology (Figure 4). At the *aspectual* level, each *Discourse Unit* markable has been tagged with three boolean attributes: *stative*, *durative* and *culminated*. At the *frame* level, *Discourse-unit* markables have been tagged with frame labels (i.e. *self_motion*, *localization*, etc.) and *Argument* markables have been assigned semantic roles (e.g. *source*, *path* or *goal*). Finally, at the *temporal*

Frame Str.	Freq.	Aspect. Str.	Freq.	Temp. Str.	Freq.
<i>self_motion</i>	739	<i>stative</i>	129	<i>initialRel.</i>	92
<i>localization</i>	114	<i>durative</i>	432	<i>ongoingRel.</i>	235
<i>vis_perc.</i>	51	<i>culminated</i>	539	<i>culminatedRel.</i>	481

Table 4: Frequencies of semantic attributes at different annotation levels.

Conn.	Freq.	S-Exp.	Freq.	Verb Form	Freq.	Verb Lex.	Freq.
<i>null</i>	494	<i>NP-VP</i>	213	<i>bare_inf</i>	480	<i>walk</i>	107
<i>and</i>	158	<i>VP</i>	691	<i>fin_pres</i>	146	<i>turn</i>	91
<i>until</i>	56			<i>gerund</i>	106	<i>continue</i>	59
<i>as</i>	37			<i>will_inf</i>	67	<i>pass</i>	57
<i>then</i>	22			<i>to_inf</i>	9	<i>null</i>	53

Table 5: Frequencies of selected LTAG-based attributes.

level, pairs of *Discourse Unit* markables corresponding to related situations have been respectively tagged as *trajector* and *landmark* and linked by a directed relation carrying a specific label (i.e. *initialRelation*, *ongoingRelation* or *subsequentRelation*). In Table 4 we provide frequencies with which respective attributes were assigned to the corresponding markables.

5.2 Grammatical Annotation

The goal of grammatical annotation was to impose the LTAG-based encoding on the texts from the corpus. Since discourse units, main verbs, arguments and connectives were labeled during the pre-processing stage, the corresponding lexical and ordering attributes became readily available, and hence were left implicit. The only explicitly tagged attribute was *verb_form*, associated with *Predicate* markables. Frequencies of selected attributes are given in Table 5.

6 Processing Model

The abstract architecture of our system comprises a set of n classification tasks $T = \{T_1, \dots, T_n\}$. Each task T_i consists in assigning a label from $L_i = \{l_{i1}, \dots, l_{im_i}\}$ to an instance representing the particular generation decision. To solve individual tasks, machine learning classifiers are trained on a set of data extracted from the annotated corpus.

Since in order to generate a well-formed and fluent text, individual generation decisions cannot be handled in isolation, the final issue to be determined during implementation, concerns the problem of integrating individual generation decisions with one another. This is a long-standing problem in NLG (cf. e.g. [Reiter, 1994]), and amounts to determining the flow of information between tasks. One advantage of casting different types of generation decisions in a single classification-oriented format, is that it is easy to implement and test them in various configurations. In the rest of this section we describe two alternative models, a sequential and an integrated one, and give the results of the initial, quantitative evaluation.

6.1 Sequential Model

In the pipeline system, implemented as a *cascade of classifiers*, the output representation is built incrementally, with subsequent classifiers having access to the outputs of the previous modules. Figure 5 illustrates this type of processing

Ontological Level	Semantic Tags	Markable Level
Aspectual Level.	<i>stative, durative, culminated</i>	<i>Discourse-unit</i>
Frame Level	<i>self_motion, visual_perception, localization</i>	<i>Discourse-unit</i>
	<i>self_mover, source, path, goal, direction, distance ...</i>	<i>Argument</i>
Temporal Level	<i>trajector, landmark</i>	<i>Discourse-unit</i>
	<i>initialRelation, ongoingRelation, subsequentRelation</i>	<i>Discourse-unit</i>

Table 3: Semantic annotation scheme

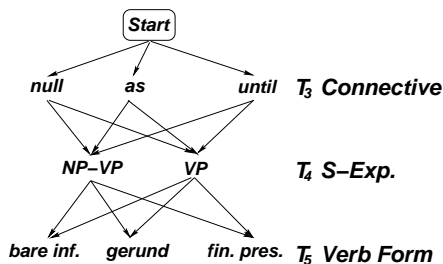


Figure 5: Sequential model on the example of three tasks: *Connective*, *S-Exp.* and *Verb Form*

as a traversal of a multi-layered lattice, with the individual layers corresponding to single tasks, and the nodes representing the respective outcomes. At each step, a corpus trained classifier outputs a probability distribution, and the transition augmented with the highest probability is selected. A well known problem with this type of processing is that generation decisions are dependant on one another and hence the initial decisions lack the necessary contextual information provided by the those occurring later (cf. [Danlos, 1984]).

6.2 Integrated Model

This problem is apparently eliminated in an integrated architecture, with all decisions being handled within a single process. As shown in Figure 6, an integrated process can be modeled as a graph, with the nodes representing outputs from individual tasks, and the edges marking inter-dependencies. The problem amounts to finding a path through the graph, so that for each task, a single node is selected, and *global* distribution of costs associated with both selected nodes and transitions is considered.

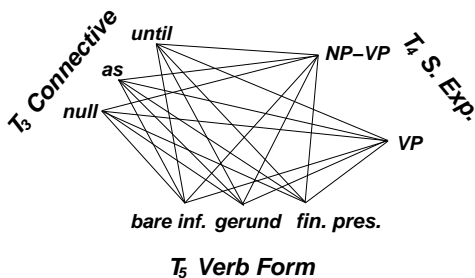


Figure 6: Integrated model

6.3 Linear Programming Formulation

To solve this graph search problem we apply an optimization technique, called Integer Linear Programming (ILP). An ILP problem consists of decision variables x_1, x_2, \dots, x_n augmented with *costs* and combined in a linear target function, and a set of constraints blocking *illegal decisions*. The goal is

to find such an assignment of the variables that the decision function is either maximized or minimized within the bounds provided by the constraints (see e.g. [Nemhauser and Wolsey, 1999]). In our application, we use *binary* variables to model both nodes in the graph, corresponding to single labels, and transitions between nodes. The objective function is then expressed as a weighted sum of the variables⁶:

$$\min c(l_{11})x(l_{11}) + c(l_{12})x(l_{12}) + \dots + c(l_{ij}l_{kp})x(l_{ij}, l_{kp})$$

The constraints we add specify that firstly, variables may take binary values only, i.e. $x \in \{0, 1\}$, secondly, for each task only one variable may be selected, and finally, if two variables $x(l_{ij})$ and $x(l_{kp})$ modeling a pair of labels belonging to two different tasks are selected, then also $x(l_{ij}, l_{kp})$ co-modeling this pair of labels must be selected⁷.

6.4 Evaluation

We evaluated both systems using *leave-one-out* cross-validation, i.e. each text was used once for testing, and the remaining texts provided the training data. We used *Naive-Bayes* classifier to learn realizations of individual tasks from the data, and to solve the ILP model we applied *lp_solve*, a GNU-licence Mixed Integer Programming (DIP) solver⁸.

The goal of the evaluation was to see how good both systems are at mirroring the human-authored texts. To assess such defined performance we applied three metrics: accuracy and *Kappa* to evaluate individual tasks and *Phi*, a distance measure that we used to compare the feature-based representations of the generated texts and the original ones. The results given in Table 6 show that both systems reached relatively high scores, with *Kappa* over 70% for almost all tasks and the end-to-end score *Phi* lying over 85%. This proves that the *meaning-to-form* mapping can be successfully learned from a relatively small corpus. In addition, an improvement of the ILP system over the pipeline in almost all tasks and the overall score *Phi* indicates that an integrated architecture offers a serious advantage over the sequential model.

7 Conclusions

In this paper we presented a corpus-based approach to building a trainable realization component for an NLG system. We concentrated on three important aspects of this task: abstract representation of the generation process in terms

⁶Costs of single nodes $c(l_{ij})$ are calculated as $-\log(P(l_{ij}))$, where $P(l_{ij})$ is the probability of selecting label l_{ij} for task T_i , output by the classifier. Costs of transitions $c(l_{ij}l_{kp})$ are given by $-\log(P(l_{ij}, l_{kp}))$, with $P(l_{ij}, l_{kp})$ denoting the joint probability of labels l_{ij} and l_{kp} co-occurring in the corpus.

⁷For details on the ILP formulation see [Marciniak and Strube, 2005a]

⁸<http://www.geocities.com/lpsolve/>

Tasks	Pipeline		ILP	
	Accuracy	κ	Accuracy	κ
<i>Adj Rank</i>	96.81%	90.90%	97.43%	92.66%
<i>Adj. Dir.</i>	98.04%	89.64%	97.95%	89.05%
<i>Connective</i>	79.10%	61.14%	79.36%	61.31%
<i>S Exp.</i>	96.20%	90.17%	99.49%	98.65%
<i>Verb Form</i>	87.83%	78.90%	93.22%	88.30%
<i>Verb Lex</i>	67.40%	64.19%	76.08%	74.00%
<i>Phrase Type</i>	87.08%	73.36%	88.03%	77.17%
<i>Prep. Lex</i>	86.95%	81.12%	88.59%	83.24%
<i>Adj Rank</i>	86.95%	78.65%	91.27%	85.72%
<i>Phi</i>	0.87		0.90	

Table 6: Results of quantitative evaluation of the ILP and pipeline systems.

of classification-oriented decisions, annotation of the corpus with the necessary linguistic information and aggregation of individual decisions within a single integrated architecture. In the evaluation we showed that a corpus-based realization in a narrow domain presents itself as a feasible task requiring a small amount of annotated data only.

Acknowledgements: The work presented here has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author receives a scholarship from KTF (09.001.2004).

References

- [Baker *et al.*, 1998] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Québec, Canada, 10–14 August 1998, pages 86–90, 1998.
- [Bangalore and Rambow, 2000a] Srinivas Bangalore and Owen Rambow. Corpus-based lexical choice in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, China 7–12 July 2000, 2000.
- [Bangalore and Rambow, 2000b] Srinivas Bangalore and Owen Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, 31 July – 4 August 2000, pages 42–48, 2000.
- [Chen *et al.*, 2002] John Chen, Srinivas Bangalore, Owen Rambow, and Marilyn Walker. Towards automatic generation of natural language generation systems. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 24 August – 1 September, 2002, 2002.
- [Danlos, 1984] Laurence Danlos. Conceptual and linguistic decisions in generation. In *Proceedings of the 10th International Conference on Computational Linguistics*, Stanford, Cal., pages 501–504, 1984.
- [Joshi and Schabes, 1991] Aravind K. Joshi and Yves Schabes. Tree-adjointing grammars and lexicalized grammars. In *Maurice Nivat and Andreas Podelski, editors, Definiteness and Recognizability of Sets of Trees*. Elsevier, 1991.
- [Kan and McKeown, 2002] M.Y. Kan and K. R. McKeown. Corpus-trained text generation for summarization. In *Proceedings of the 2nd International Conference on Natural Language Generation*, New York, NY, 1-3 July, 2002, 2002.
- [Langkilde and Knight, 1998] Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Québec, Canada, 10–14 August 1998, pages 704–710, 1998.
- [Lapata, 2003] Mirella Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, pages 545–552, 2003.
- [Marciniak and Strube, 2005a] Tomasz Marciniak and Michael Strube. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, Ann Arbor, MI, 29-30 June, 2005, pages 136–143, 2005.
- [Marciniak and Strube, 2005b] Tomasz Marciniak and Michael Strube. Modeling and annotating the semantics of route directions. In *Proceedings of the 6th International Workshop on Computational Semantics*, Tilburg, The Netherlands, January 12-14, 2005, pages 151–162, 2005.
- [Moens and Steedman, 1988] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28, 1988.
- [Nemhauser and Wolsey, 1999] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley, New York, NY, 1999.
- [Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK, 2000.
- [Reiter and Sripada, 2002] Ehud Reiter and Somayajulu Sripada. Should corpora texts be gold standards for NLG? In *Proceedings of the 2nd International Conference on Natural Language Generation*, New York, NY, 1-3 July, 2002, pages 97–104, 2002.
- [Reiter *et al.*, 2003] Ehud Reiter, Somayajulu Sripada, and Roma Robertson. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research*, 18:491–516, 2003.
- [Reiter, 1994] Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport, MA, 21-24 June 1994, pages 160–173, 1994.
- [Shaw and Hatzivassiloglou, 1999] James Shaw and Vasileios Hatzivassiloglou. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, July 1999, pages 135–143, 1999.
- [Varges and Mellish, 2001] Sebastian Varges and Chris Mellish. Instance-based natural language generation. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA, 2-7 June, 2001, pages 1–8, 2001.
- [Vendler, 1967] Zeno Vendler. Verbs and times. In *Linguistics in Philosophy*, Cornell University Press, Ithaca, NY, pages 97–121, 1967.
- [Webber and Joshi, 1998] Bonnie Lynn Webber and Aravind Joshi. Anchoring a lexicalized tree-adjointing grammar for discourse. In *Proceedings of the COLING/ACL '98 Workshop on Discourse Relations and Discourse Markers*, Montréal, Québec, Canada, 15 August 1998, pages 86–92, 1998.