Discrete Optimization as an Alternative to Sequential Processing in NLG

Tomasz Marciniak and Michael Strube
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

We present an NLG system that uses Integer Linear Programming to integrate different decisions involved in the generation process. Our approach provides an alternative to pipeline-based sequential processing which has become prevalent in today's NLG applications.

1 Introduction

From an engineering perspective, one of the major considerations in building a Natural Language Generation (NLG) system is the choice of the architecture. Two important issues that need to be considered at this stage are firstly, the *modularization* of the linguistic decisions involved in the generation process and secondly, the *processing flow* (cf. [De Smedt *et al.*, 1996]).

On one side of the spectrum lie integrated systems, with all linguistic decisions being handled within a single process (e.g. [Appelt, 1985]). Such architectures are theoretically attractive, as they assume a close coordination of different types of linguistic decisions, which are known to be dependent on one another (cf. e.g. [Danlos, 1984]). A major disadvantage of integrated models is the complexity that they necessarily involve, which results in poor portability and scalability. On the other side of the spectrum there are highly modularized pipeline architectures. A prominent example of this second case is the *consensus* pipeline architecture recognized by [Reiter, 1994] and further elaborated in [Reiter and Dale, 2000]. The modularization of Reiter's model occurs at two levels. First, individual linguistic decisions of the same type (e.g. involving *lexical* or *syntactic* choice) are grouped together within single low level tasks, such as *lexicalization*, *aggregation* or *ordering*. Second, tasks are allocated to three high-level generation stages, i.e. *Document Planning*, *Microplanning* and *Surface Realization*. The processing flow in the pipeline architecture is sequential, with individual tasks being executed in a predetermined order.

A study of applied NLG systems [Cahill and Reape, 1999] reveals, however, that while most applied NLG systems rely on sequential processing, they do not follow the strict modularization that the consensus model assumes. Low-level tasks are spread over various generation stages and may in fact be executed more than once at diverse positions in the pipeline.

An attempt to account for commonalities that many NLG systems share, without imposing too many restrictions, as is the case with Reiter's "consensus" model, is the Reference Architecture for Generation Systems (RAGS) [Mellish *et al.*, 2004]. RAGS is an abstract specification of an NLG architecture that focuses on two issues: data types that the generation process manipulates and a generic model of the interactions between modules, based on a common *central server*. An important feature of RAGS is that it leaves the question of processing flow to the actual implementation. Hence it is theoretically possible to build both fully integrated as well as pipeline-based systems that would observe the RAGS principles. Two implementations of RAGS presented in [Mellish and Evans, 2004] demonstrate an intermediate way.

In this paper we present a novel approach to building an integrated NLG system, in which the generation process is modeled as a discrete optimization problem. It provides an extension to the *classification-based generation* framework, presented in [Marciniak and Strube, 2004]. We first assume modularization of the generation process at the lowest possible level: individual tasks correspond to realizations of single form elements (FEs) that build up a linguistic expression. The decisions that these tasks involve are then represented as classification tasks and integrated via an Integer Linear Programming (ILP) formulation (see e.g. [Nemhauser and Wolsey, 1999]. This way we avoid the well known ordering problem that is present in all pipeline-based systems. Observing, at least partially, the methodological principles of RAGS, we specify the architecture of our system at two independent levels. At the abstract level, the low-level generation tasks are defined, all based on the same input/output interface. At the implementation level, the processing flow and integration method are determined.

The rest of the paper is organized as follows: in Section 2 we present briefly the classification-based generation framework and remark on the shortcomings of pipeline-based processing. In Section 3 we introduce the ILP formulation of the generation task, and in Section 4 we report on the experiments and evaluation of the system.

2 Classification-Based Generation

In informal terms, classification can be characterized as the task of assigning a class label to an unknown instance, given a set of its properties and attributes represented as a feature vec-

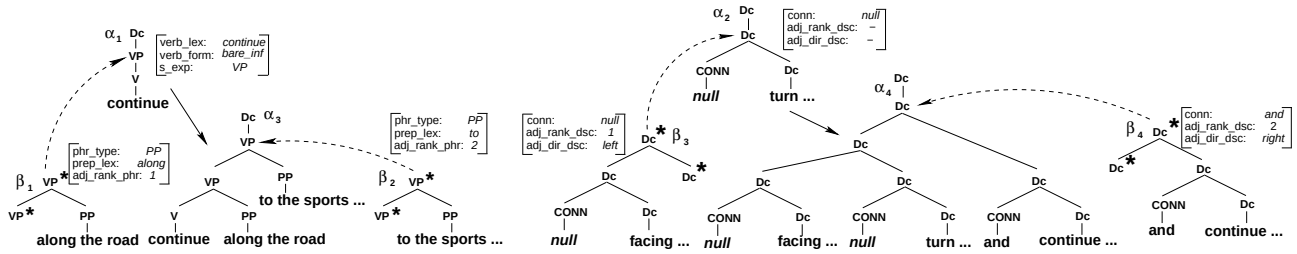

Figure 1: LTAG-based derivation at the clause (left) and discourse levels (right). Elementary trees are represented as feature vectors. Adjunction operations are marked with dashed arrows.

tor. In recent years supervised machine learning methods relying on pre-classified *training data* have been applied in various areas of NLP to solve tasks formulated as classification problems. In NLG machine learning methods have been used to solve single tasks such as content selection and ordering (e.g. [Duboue, 2004; Dimitromanolaki and Androutsopoulos, 2003]), lexicalization (e.g. [Reiter and Sripada, 2004]) and referring expressions generation (e.g. [Cheng *et al.*, 2001]).

In these applications classifiers trained on labeled data have proven more robust and efficient than approaches using explicit expert knowledge. The difficulty of formalizing the linguistic knowledge involved in the development of a knowledge-based system (a.k.a. *knowledge-acquisition-bottleneck*) has been replaced with an effort of obtaining the right kind of data, which typically involves annotating manually a corpus of relevant texts with the required linguistic information (cf. [Daelemans, 1993]).

The classification-based generation framework that we introduced in [Marciniak and Strube, 2004] is based on a simple idea that the linguistic form of an expression can be decomposed into a set of discrete form elements (FEs) representing both its syntactic and lexical properties. The generation process is then modeled as a series of classification tasks that realize individual FEs. Realization of each FE is then regarded as a single low-level generation task.

2.1 Route Directions

As the main application for this work we consider the task of generating natural language *route directions*. An example of such a text is given below:

> (a) Facing the Wildcat statue, (b) turn left on the brick sidewalk (c) and continue along the road to the Sports Complex. (d) Make a right onto Concord Road, (e) and keep going straight, (f) passing Presbyterian Church on your left, (g) until you reach Copeland Street. (h) The library building will be just around the corner on your right.

We analyze the content of instructional texts of this kind in terms of temporally related situations, i.e. *actions* (b, c, d, e) *states* (a, h) and *events* (f, g), denoted by individual discourse units. The temporal structure of the texts is then modeled as a tree, with nodes representing individual situation descriptions and edges signaling the relations (see Figure 2). The semantics of each discourse unit is further represented as a feature vector describing the aspectual category and frame structure of the profiled situation. This tree-based representation of the semantic content of route directions constitutes the input to the generation process. A detailed description of the underlying conceptual model and the annotation process is presented in [Marciniak and Strube, 2005].

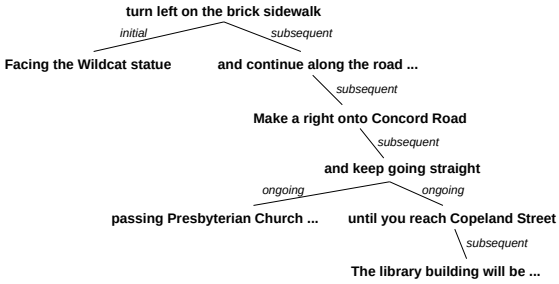

Figure 2: Temporal Structure

2.2 From LTAG to Form Elements

To specify an inventory of FEs that would become objects of the low-level generation tasks, we first apply the Lexicalized Tree Adjoining Grammar (LTAG) formalism (see e.g. [Joshi and Schabes, 1991]) to model the linguistic form of the texts. In LTAG, the derivation of a linguistic structure starts with a selection of *elementary* trees, anchored by lexical items, such as verbs or prepositions at the clause level and discourse connectives at the discourse level (cf. [Webber and Joshi, 1998]). In the next step, elementary trees are put together by means of *adjunction* operations that follow the dependency structure provided by the *derivation tree*. We take the temporal structure from Figure 2 to constitute the discourse level derivation tree, with the temporal relationships corresponding to the syntactic dependencies. At the clause level, the derivation tree is isomorphic with the frame-based ontological representation of individual situations (see [Marciniak and Strube, 2005]).

The clause- and discourse-level derivation of discourse unit (c) from the above example in the context of (a) and (b) is depicted in Figure 1. At the clause level, the set of elementary trees includes one *initial* tree α_1 anchored by the main verb, which also specifies the syntactic frame of the clause, and *auxiliary* trees β_1 and β_2 corresponding to the verb arguments. At the discourse level, the discourse unit which occupies the root position in the temporal structure (cf. Figure 2)

Adj. Rank	*Adj. Dir.*	*Conn.*	*S Exp.*	*Verb Lex.*	*Verb Form*
1	right	*and*	VP	*continue*	Bare Inf.
Phr. Type$_1$	*Prep. Lex.*$_1$	*Adj. Rank*$_1$	*Phr. Type*$_2$	*Prep. Lex.*$_2$	*Adj. Rank*$_2$
PP	*along*	1	PP	*to*	2

Table 1: FEs based form representation of *and continue along the road to the sport complex*.

is modeled as the initial tree α_2, and auxiliary trees β_3 and β_4 represent the remaining discourse units.

To model the whole process in a uniform way we encode the elementary trees as feature vectors, with individual features conveying syntactic (e.g. *s_exp*) and lexical (e.g. *verb_lex*) information. Features *adj_rank* and *adj_dir* denote respectively the ordering of the adjunction operations and the adjunction direction, which both determine the linear structure of the text. Hence the form of the whole discourse can be represented in terms of feature-value pairs used to encode the initial trees and the derivation process. On that basis we define a set of form elements building up a discourse as directly corresponding to the individual features. A detailed description of the FEs is given below:

FE$_1$: **Adjunction Rank / Disc. Level** specifies the linear rank of each discourse unit at the local level, i.e. only clauses temporally related to the same *parent* clause are considered.

FE$_2$: **Adjunction Direction** is concerned with the position of the *child* discourse unit relative to the parent one (e.g. (a) *left of* (b), (c) *right of* (b), etc.).

FE$_3$: **Connective** determines the lexical form of the discourse connective (e.g. *null* in (a), *until* in (g)).

FE$_4$: **S Expansion** specifies whether a given discourse unit is realized as a clause with the explicit subject (i.e. np+vp *expansion* of the root S node in a clause) (e.g. (g, h)) or not (e.g. (a), (b)).

FE$_5$: **Verb Form** denotes the form of the main verb in a clause (e.g. *gerund* in (a), (c), *bare infinitive* in (b), *finite present* in (g), etc.).

FE$_6$: **Verb Lex.** specifies the lexical form of the main verb (e.g. *turn* in (b), *pass* in (f) or *reach* in (g)).

FE$_7$: **Phrase Type** determines for each argument in a clause its syntactic realization as a *noun phrase* (NP) , *prepositional phrase* (PP) or a *particle* (P).

FE$_8$: **Preposition Lex.** is concerned with the choice of a lexical form for prepositions or particles in argument phrases (e.g. *left* and *on* in (b) or *along* and *to* in (c)). If the value of FE_7 is NP, then this FE is set to *none*.

FE$_9$: **Adjunction Rank / Phr. Level** specifies the linear rank of each verb argument within a clause.

As an example, consider the FEs-based representation of the form of clause (c) presented in Table 1. Realization of each FE_i is represented as a classification task T_i, with a set of possible *class labels* corresponding to the different forms that FE_i may take. Only tasks T_1 and T_9 associated respectively with *Adjunction Rank / Disc. Level* and *Adjunction Rank / Phr. Level* are split into a series of binary *precedence* classifications that determine the relative position of two discourse units or phrasal arguments at a time (e.g. (a) $\prec$ (c), (c) $\prec$ (d), and similarly *along the road* $\prec$ *to the sports complex* etc.). These partial results are later combined to determine the rank of the respective constituents.

Arguably, the above FEs and the corresponding tasks are independent of the underlying grammatical model. In this work we use the abstraction of the grammatical structure provided by LTAG, but the same or a similar set of FEs can be readily derived from other formalisms (cf. e.g. [Meteer, 1990]). The role of the grammatical theory in defining form elements is twofold. First, it specifies the exact position of individual FEs in the grammatical structure, making it clear how they should be assembled. Second, it ensures a wide coverage: although the linguistic structures that we consider here are relatively simple, the use of LTAG as the underlying grammatical formalism guarantees that our generation framework can be applied to producing much more complex constructions, both at the clause and discourse levels. Apparently, this would require a richer feature vector representation of the initial trees, and hence a larger number of FEs and the corresponding generation tasks. The basic principles of the generation process, however, would remain unchanged.

Notice also that the tasks considered here can be grouped under the conventional NLG labels, such as *text structuring* (i.e. T_1, T_2), *lexicalization* (i.e. T_3, T_6, T_8) and *sentence realization* (i.e. T_4, T_5, T_9). Yet another important NLG task, i.e. *aggregation* appears to be handled indirectly by T_3 (e.g. *Turn left. Continue along the road.* vs. *Turn left and continue along the road.*) and T_5 (e.g. *Keep going straight. You will pass the Presbyterian Church on your right.* vs. *Keep going straight, passing the Presbyterian Church on your right.*). We view it as the strength of our approach that regardless of their different linguistic character all these tasks are modeled in *exactly* the same way.

2.3 System Architecture and Sequential Processing

At an abstract level, the architecture of our system consists of an unordered set of classifiers solving individual generation tasks. Each classifier is trained on a separate set of data obtained from the corpus of route directions annotated with both semantic and grammatical information.

In the previous work [Marciniak and Strube, 2004] we followed the sequential paradigm advocated by [Daelemans and van den Bosch, 1998] and implemented the system as a *cascade of classifiers*. In such systems the output representation is built incrementally, with subsequent classifiers having access to the outputs of previous modules. An important characteristic of this model is its extensibility. Since classifiers rely on a uniform representation of the input (i.e. a feature vector) and the output (i.e. a single feature value), it is easy to change the ordering or insert new modules at any place in the pipeline. Both operations only require retraining classifiers with a new selection of the input features.

A major problem that we faced was that we found no satisfactory method to determine the *right* ordering of individual classifiers that would guarantee optimal realization of the grammatical form of the generated expression. We found out that no matter what ordering we adopted tasks that were solved at the begining had a lower accuracy as the necessary contextual information, i.e. based on the outcomes from other tasks, was missing. At the same time, subsequent

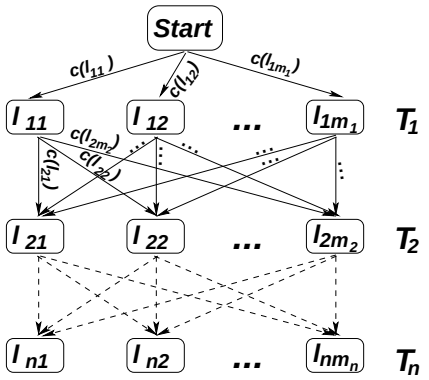

Figure 3: Sequential processing as a graph.

tasks were influenced by the initial decisions, which in some cases led to *error propagation*. Apparently, this was due to the well known fact that elements of the linguistic structure are strongly correlated with one another (see e.g. [Danlos, 1984]). Hence individual generation decisions should not be handled in isolation and arranging them in a fixed order will always involve a specific *ordering bias*.

To get a feeling for the limitations that sequential processing of generation tasks involves, consider its graphical representation in Figure 3. The process corresponds to the *best-first* traversal of a weighted multi-layered lattice. Separate layers $T_1, ..., T_n$ correspond to the individual tasks, and the nodes at each layer $(l_{i1}, ..., l_{im_i})$ represent class labels for each task[1]. In the sequential model only transitions between nodes belonging to subsequent layers are granted. Each such transition is augmented with a transition cost, which may be affected by the traversal *history* but does not consider the future choices. Nodes selected in this process represent the outcomes of individual tasks. As can be seen, the process is *locally* driven and it does not guarantee an optimal realization of the tasks.

As an example consider three interrelated form elements: Connective, S Exp. and Verb Form and their different realizations presented in Table 2. Apparently each of these FEs has the potential to affect the overall meaning of the discourse unit or its stylistics. It can also be seen that only certain combinations of different forms are allowed in the given semantic context. Different realization of any of these FEs would require other elements to be changed accordingly. To conclude, following Danlos' observation, we see no *a priori* reason to impose any fixed ordering on the respective generation tasks, and the experiments that we describe in Section 4 support this position.

3 Discrete Optimization Model

As an alternative to sequential ordering of the generation tasks we consider the *metric labeling problem* formulated by [Kleinberg and Tardos, 2000], and originally applied in an

[1]Since different generation tasks may have varying numbers of labels we denote the cardinality of L_i, i.e. the set of possible labels for task T_i, as m_i.

Discourse Unit	FE_3	FE_4	FE_5
Pass the First Union Bank ...	null	vp	bare inf.
It is necessary that you pass ...	null	np+vp	bare inf.
Passing the First Union Bank ...	null	vp	gerund
After passing the First Union Bank ...	after	vp	gerund
After your passing ...	after	np+vp	gerund
As you pass the First Union Bank ...	as	np+vp	fin. pres.
Until you pass the First Union Bank ...	until	np+vp	fin. pres.
Until passing ...	until	vp	gerund

Table 2: Different realizations of form elements: Connective, Verb Form and S Expansion. Rare but correct constructions are in italics.

image restoration application, where classifiers determine the "true" intensity values of individual pixels. This task is formulated as a labeling function $f : P \rightarrow L$ which maps a set P of n objects onto a set L of m possible labels. The goal is to find an assignment that minimizes the overall cost function $Q(f)$ which has two components: *assignment costs*, i.e. the costs of selecting a particular label for individual objects, and *separation costs*, i.e. the costs of selecting a pair of labels for two *related* objects[2]. [Chekuri *et al.*, 2001] proposed an integer linear programming (ILP) formulation of the metric labeling problem, with both assignment cost and separation costs being modeled as binary variables of the linear cost function.

Recently, [Roth and Yih, 2004] applied an ILP model to the task of the simultaneous assignment of semantic roles to the entities mentioned in a sentence and recognition of the relations holding between them. The assignment costs were calculated on the basis of predictions of *basic* classifiers, i.e. trained for both tasks individually with no access to the outcomes of the other task. The separation costs were formulated in terms of binary constraints which specified whether a specific semantic role could occur in a given relation, or not.

In the remainder of this paper, we present a more general model, which we apply to the generation tasks presented in Section 2. We put no limits on the number of tasks being solved, and express the separation costs as stochastic constraints, which can be calculated off-line from the available linguistic data.

3.1 ILP Formulation

We consider a general context in which the generation process comprises a range of linguistic decisions modeled as a set of n classification tasks $T = \{T_1, ..., T_n\}$ which potentially form mutually related pairs.

Each task T_i consists in assigning a label from $L_i = \{l_{i1}, ..., l_{im_i}\}$ to an instance that represents the particular decision. Assignments are modeled as variables of a linear cost function. We differentiate between *simple variables* that model individual assignments of labels and *compound variables* that represent respective assignments for each pair of related tasks.

To represent individual assignments the following procedure is applied: for each task T_i, every label from L_i is asso-

[2]These costs were calculated as the function of the *metric* distance between a pair of pixels and the difference in intensity.

ciated with a binary variable $x(l_{ij})$. Each such variable represents a binary choice, i.e. a respective label l_{ij} is selected if $x(l_{ij}) = 1$ or rejected otherwise. The coefficient of variable $x(l_{ij})$ which models the assignment cost $c(l_{ij})$ is given by:

$$c(l_{ij}) = -log_2(p(l_{ij}))$$

where $p(l_{ij})$ is the probability of l_{ij} being selected as the outcome of task T_i. The probability distribution for each task is provided by the basic classifiers that do not consider the outcomes of other tasks[3].

The role of compound variables is to provide pairwise constraints on the outcomes of individual tasks. Since we are interested in constraining only those tasks are that truly dependent on one another we first apply the contingency coefficient C to measure the degree of correlation for each pair of tasks[4]. In the case of tasks T_i and T_k which are *significantly correlated*, for each pair of labels from $L_i \times L_k$ we build a single variable $x(l_{ij}, l_{kp})$. Each such variable is associated with a coefficient representing the constraint on the respective pair of labels l_{ij}, l_{kp} calculated in the following way:

$$c(l_{ij}, l_{kp}) = -log_2(p(l_{ij,}l_{kp}))$$

with $p(l_{ij,}l_{kp})$ denoting the *prior* joint probability of labels l_{ij} and l_{kp} in the data, which is independent from the general classification context and hence can be calculated off-line[5].

The ILP model consists of the target function and a set of constraints which block *illegal* assignments (e.g. only one label of the given task can be selected)[6]. In our case the target function is the cost function $Q(f)$, which we want to minimize:

$$min \;\; Q(f) = \sum_{T_i \in T} \sum_{l_{ij} \in L_i} c(l_{ij}) \cdot x(l_{ij})$$

$$+ \sum_{T_i, T_k \in T, i<k} \sum_{<l_{ij}, l_{kp}> \in L_i \times L_k} c(l_{ij}, l_{kp}) \cdot x(l_{ij}, l_{kp})$$

Constraints need to be formulated for both the simple and compound variables. First we want to ensure that exactly one label l_{ij} belonging to task T_i is selected, i.e. only one simple variable $x(l_{ij})$ representing labels of a given task can be set to 1:

$$\sum_{l_{ij} \in L_i} x(l_{ij}) = 1, \qquad \forall i \in \{1, ..., n\}$$

[3]In this case the ordering of tasks is not necessary, and the classifiers can run independently from each other.

[4]C is a test for measuring the association of two nominal variables, and hence adequate for the type of tasks that we consider here. The coefficient takes values from 0 (no correlation) to 1 (complete correlation) and is calculated by the formula: $C = (\chi^2/(N + \chi^2))^{1/2}$, where χ^2 is the chi-squared statistic and N the total number of instances. The significance of C is then determined from the value of χ^2 for the given data. See e.g. [Goodman and Kruskal, 1972].

[5]In Section 4 we discuss an alternative approach which considers the actual input.

[6]For a detailed overview of linear programming and different types of LP problems see e.g. [Nemhauser and Wolsey, 1999].

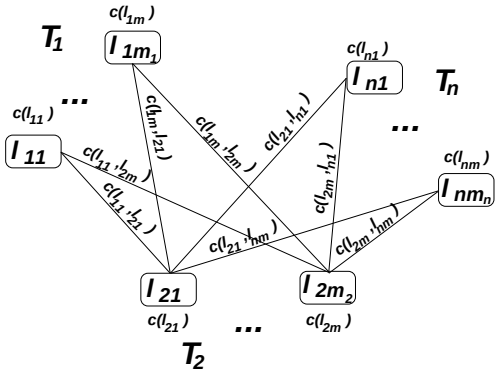

Figure 4: Graph representation of the ILP model.

We also require that if two simple variables $x(l_{ij})$ and $x(l_{kp})$, modeling respectively labels l_{ij} and l_{kp}, are set to 1, then the compound variable $x(l_{ij}, l_{kp})$, which models co-occurrence of these labels, is also set to 1. This is done in two steps: we first ensure that if $x(l_{ij}) = 1$, then exactly one variable $x(l_{ij}, l_{kp})$ must also be set to 1:

$$x(l_{ij}) - \sum_{l_{kp} \in L_k} x(l_{ij}, l_{kp}) = 0,$$

$$\forall i, k \in \{1, ..., n\}, i < k \;\wedge\; j \in \{1, ..., m_i\}$$

and do the same for variable $x(l_{kp})$:

$$x(l_{kp}) - \sum_{l_{ij} \in L_i} x(l_{ij}, l_{kp}) = 0,$$

$$\forall i, k \in \{1, ..., n\}, i < k \;\wedge\; p \in \{1, ..., m_k\}$$

Finally, we constrain the values of both simple and compound variables to be binary:

$$x(l_{ij}) \in \{0, 1\} \;\wedge\; x(l_{ij}, l_{kp}) \in \{0, 1\},$$

$$\forall i, k \in \{1, ..., n\} \;\wedge\; j \in \{1, ..., m_i\} \;\wedge\; p \in \{1, ..., m_k\}$$

We can represent the decision process that our ILP model involves as a graph, with the nodes corresponding to individual labels and the edges marking the associations between labels belonging to correlated tasks. In Figure 4, task T_1 is correlated with task T_2 and task T_2 with task T_n. No correlation exists for pair T_1, T_n. Both nodes and edges are augmented with costs. The goal is to select a subset of connected nodes, minimizing the overall cost, given that for each group of nodes $T_1, T_2, ..., T_n$ exactly one node must be selected, and the selected nodes, representing correlated tasks, must be connected. We can see that in contrast to the pipeline approach (cf. Figure 1), no *local* decisions determine the overall assignment as the *global* distribution of costs is considered.

4 Experiments and Results

In order to evaluate our approach we conducted a series of experiments with two implementations of the ILP model and two different pipelines. Each system takes as input the tree-based representation of the semantic content of route directions described in Section 2. The generation process traverses the temporal tree in a depth-first fashion, and for each node a single discourse unit is realized.

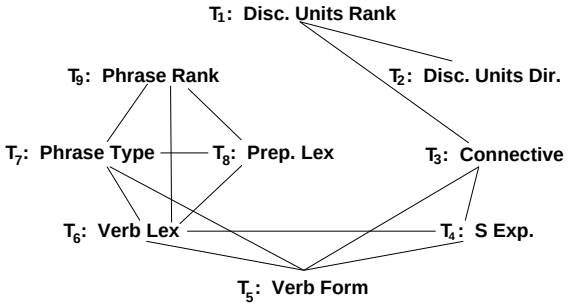

Figure 5: Correlation network for the generation tasks.

null	*and*	*as*	*after*	*until*	T_3 Connective / T_5 Verb Form
0.40	0.18	0	0	0	*bare_inf*
0	0	0	0.04	0.01	*gerund*
0.05	0.01	0.06	0.03	0.06	*fin_pres*
0.06	0.05	0	0	0	*will_inf*

Table 3: Joint distribution matrix for selected labels of tasks Connective *(horizontal)* and Verb Form *(vertical)*, computed for all discourse units in a corpus.

4.1 Correlations Between Tasks

We started with running the correlation tests for all pairs of tasks. The obtained correlation network is presented in Figure 5. It is interesting to observe that tasks which realize FEs belonging to the same levels of linguistic organization, and have traditionally been handled within the same generation stages (i.e. *Text Planning*, *Microplanning* and *Realization*) are closely correlated with one another. This fact supports empirically some assumptions behind Reiter's consensus model. On the other hand, there exist quite a few correlations that extend over the stage boundaries, and all three lexicalization tasks i.e. T_3, T_6 and T_8 are correlated with many tasks of a totally different linguistic character.

4.2 ILP Systems

We used the ILP model described in Section 3 to implement two generation systems. To obtain assignment costs, both systems get a probability distribution for each task from basic classifiers trained on the training data. To calculate the separation costs, modeling the stochastic constraints on the co-occurrence of labels, we considered correlated tasks only (cf. Figure 5) and applied two calculation methods, which resulted in two different system implementations.

In ILP1, for each pair of tasks we computed the joint distribution of the respective labels considering all discourse units in the training data before the actual input was known. Such obtained joint distributions were used for generating all discourse units from the test data. An example matrix with joint distribution for selected labels of tasks Connective and Verb Form is given in Table 3. An advantage of this approach is that the computation can be done in an *offline* mode and has no impact on the run-time.

In ILP2, the joint distribution for a pair of tasks was calculated at run-time, i.e. only after the actual input had been known. This time we did not consider all discourse units in the training data, but only those whose meaning, represented as a feature vector, was *similar* to the meaning of the input discourse unit. As a similarity metric we used the *Phi* coefficient[7], and set the similarity threshold at 0.8. As can be seen from Table 4, the probability distribution computed in this way is better suited to the specific semantic context. This is especially important if the available corpus is small and the frequency of certain pairs of labels might be too low to have a significant impact on the final assignment.

null	*and*	*as*	*after*	*until*	T_3 Connective / T_5 Verb Form
0.13	0.02	0	0	0	*bare_inf*
0	0	0	0	0	*gerund*
0	0	0.05	0.02	0.27	*fin_pres*
0.36	0.13	0	0	0	*will_inf*

Table 4: Joint distribution matrix for tasks Connective and Verb Form, considering only disc. units similar to (c): *until you see the river side in front of you*, at *Phi*-threshold ≥ 0.8.

4.3 Pipeline Systems

As a baseline we implemented two pipeline systems. In the first one we used the ordering of tasks that resembles most closely the standard NLG pipeline and which we also used before in [Marciniak and Strube, 2004][8].

Individual classifiers had access to both the semantic features, and the features output by the previous modules. To train the classifiers, the *correct* feature values were extracted from the training data and during testing the *generated*, and hence possibly erroneous, values were taken.

In the other pipeline system we wanted to minimize the error-propagation effect and placed the tasks in the order of *decreasing* accuracy. To determine the ordering of tasks we applied the following procedure: the classifier with the highest baseline accuracy was selected as the first one. The remaining classifiers were trained and tested again, but this time they had access to the additional feature. Again, the classifier with the highest accuracy was selected and the procedure was repeated until all classifiers were ordered.

4.4 Evaluation

We evaluated our system using *leave-one-out* cross-validation, i.e. for all texts in the corpus, each text was used once for testing, and the remaining texts provided the training data. To solve individual classification tasks we used the decision tree learner *C4.5* in the pipeline systems and the *Naive Bayes* algorithm[9] in the ILP systems. Both learning schemes

[7] *Phi* is a measure of the extent of correlation between two sets of binary variables, see e.g. [Edwards, 1976]. To represent multi-class features on a binary scale we applied *dummy coding* which transforms multi class-nominal variables to a set of dummy variables with binary values.

[8] The ordering of tasks is given in Table 5.

[9] Both implemented in the Weka machine learning software [Witten and Frank, 2000].

Tasks	Pipeline 1 Pos.	Pipeline 1 Accuracy	Pipeline 1 κ	Pipeline 2 Pos.	Pipeline 2 Accuracy	Pipeline 2 κ	ILP 1 Accuracy	ILP 1 κ	ILP 2 Accuracy	ILP 2 κ
Dis.Un. Rank	1	96.81%	90.90%	2	96.81%	90.90%	97.43%	92.66%	97.43%	92.66%
Dis.Un. Pos.	2	98.04%	89.64%	1	98.04%	89.64%	96.10%	77.19%	97.95%	89.05%
Connective	3	78.64%	60.33%	8	79.10%	61.14%	79.15%	61.22%	79.36%	61.31%
S Exp.	4	95.90%	89.45%	3	96.20%	90.17%	99.48%	98.65%	99.49%	98.65%
Verb Form	5	86.76%	77.01%	4	87.83%	78.90%	92.81%	87.60%	93.22%	88.30%
Verb Lex.	6	64.58%	60.87%	9	67.40%	64.19%	75.87%	73.69%	76.08%	74.00%
Phr. Type	7	86.93%	75.07%	5	87.08%	75.36%	87.33%	76.75%	88.03%	77.17%
Prep. Lex.	8	86.23%	81.12%	6	86.03%	81.10%	87.28%	82.20%	88.59%	83.24%
Phr. Rank	9	84.73%	75.24%	7	86.95%	78.65%	90.22%	84.02%	91.27%	85.72%
Phi		0.85			0.87		0.89		0.90	

Table 5: Results reached by the implemented ILP systems and two baselines. For both pipeline systems, *Pos.* stands for the position of the tasks in the pipeline.

yielded highest results in the respective configurations[10]. To solve the ILP models we used *lp_solve*, a highly efficient GNU-licence Mixed Integer Programming (MIP) solver[11], that implements the *Branch-and-Bound* algorithm. For each task we applied a *feature selection* procedure (cf. [Kohavi and John, 1997]) to determine which *semantic* features should be taken as the input by the basic classifiers.

To evaluate individual tasks we applied two metrics: accuracy, calculated as the proportion of correct classifications to the total number of instances, and the κ statistic, which corrects for the proportion of classifications that might occur by chance. For *end-to-end* evaluation, we applied the *Phi* coefficient to measure the degree of similarity between the vector representations of the generated form (i.e. built from the outcomes of individual tasks) and the reference form obtained from the test data. The *Phi*-based similarity metric is similar to κ as it compensates for the fact that a match between two multi-label features is more difficult to obtain than in the case of binary features. This measure tells us how well all the tasks have been solved together, which in our case amounts to generating the whole text.

The results presented in Table 5 show that the ILP systems achieved highest accuracy and κ for most tasks and reached the highest overall *Phi* score. Notice that ILP2 improved the accuracy of both pipeline systems for the three correlated tasks that we discussed before, i.e. Connective, S Exp. and Verb Form. Another group of correlated tasks for which the results appear interesting are i.e. Verb Lex., Phrase Type and Phrase Rank (cf. Figure 3). Notice that Verb Lex. got higher scores in Pipeline2, with outputs from both Phrase Type and Phrase Rank (see the respective pipeline positions), but the reverse effect did not occur: scores for both phrase tasks were lower in Pipeline1 when they had access to the output from Verb Lex., contrary to what we might expect. Apparently, this was due to the low accuracy for Verb Lex. which caused the already mentioned error propagation. This example shows well the advantage that optimization processing brings: both ILP systems reached much higher scores for all three tasks.

Finally, it appears as no coincidence that the three tasks involving lexical choice, i.e. Connective, Verb Lex. and Preposition Lex. scored lower than the syntactic tasks in all systems. This can be attributed partially to the limitations of retrieval measures which do not allow for the fact, that in a given semantic content more than one lexical form can be appropriate.

5 Conclusions

In this paper we showed that the pipeline architecture in an NLG application can be successfully replaced with an integrated ILP-based model which is better suited to handling correlated generation decisions. To the best of our knowledge, linear programming has been used in an NLG related work only by [Althaus *et al.*, 2004] to solve a single task of determining the order of discourse constituents. In a somewhat related context [Dras, 1999] used ILP to optimize the task of text paraphrasing, given global constraints such as text and sentence length, readibilty, etc.

In contrast, in this work we use an ILP model to organize the entire process of generating the surface form from an underlying semantic representation, which involves an integration of different types of NLG tasks. Although in our system we use machine learning as the primary decision making mechanism, we believe that the ILP model can also be used with knowledge-based systems that observe the classification-oriented formulation of the NLG tasks.

Finally, we are convinced that an adequate evaluation of an NLG system must at some stage go beyond the application of quantitative measures. Nevertheless, it is reasonable to expect that the improvement that we reached with the ILP system, especially the increase of the overall *Phi* score, must correlate to some extent with the quality improvement. To verify it we are currently proceeding with qualitative evaluation of the output from our system.

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).

[10] We have found that in direct comparison *C4.5* performs better than *Naive Bayes* but the probability distribution that it outputs is strongly biased towards the *winning* label. In this case it is practically impossible for the ILP system to change the classifier's decision, as the costs of other labels get extremely high. Hence the more balanced probability distribution given by *Naive Bayes* can be easier *corrected* in the optimization process.

[11] http://www.geocities.com/lpsolve/

References

[Althaus *et al.*, 2004] Ernst Althaus, Nikiforos Karamanis, and Alexander Koller. Computing locally coherent discourses. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, July 21-26, 2004, pages 399–406, 2004.

[Appelt, 1985] Douglas Appelt. *Planning English Sentences*. Cambridge University Press, Cambridge, UK, 1985.

[Cahill and Reape, 1999] Lynne Cahill and Mike Reape. Component tasks in applied NLG systems. Technical Report ITRI-99-05, ITRI, University of Brighton, March 1999.

[Chekuri *et al.*, 2001] Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the 12th Annual ACM SIAM Symposium on Discrete Algorithms*, Washington, DC, pages 109–118, 2001.

[Cheng *et al.*, 2001] Hua Cheng, Massimo Poesio, Renate Henschel, and Chris Mellish. Corpus-based NP modifier 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 9–16, 2001.

[Daelemans and van den Bosch, 1998] Walter Daelemans and Antal van den Bosch. Rapid development of NLP modules with memory-based learning. In *Proceedings of ELSNET in Wonderland. Utrecht: ELSNET*, pages 105–113, 1998.

[Daelemans, 1993] Walter Daelemans. Memory-based lexical acquisition and processing. In *Proceedings of the Third International EAMT Workshop on Machine Translation and the Lexicon*, Heidelberg, Germany, 26-28 April, 1993, pages 85–98, 1993.

[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.

[De Smedt *et al.*, 1996] Koenraad De Smedt, Helmut Horacek, and Michael Zock. Architectures for natural language generation: Problems and perspectives. In G. Adorni and M. Zock, editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, pages 17–46. Springer Verlag, 1996.

[Dimitromanolaki and Androutsopoulos, 2003] Aggeliki Dimitromanolaki and Ion Androutsopoulos. Learning to order facts for discourse planning in natural language generation. In *Proc. of the 9th European Workshop on Natural Language Generation*, Budapest, Hungary, 13 – 14 April 2003, pages 23–30, 2003.

[Dras, 1999] Mark Dras. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. PhD thesis, Macquarie University, Australia, 1999.

[Duboue, 2004] Pablo A. Duboue. Indirect supervised learning of content selection logic. In *Proceedings of the 3rd International Conference on Natural Language Generation*, Brockenhurst, UK, 14-16 July, 2004, pages 41–50, 2004.

[Edwards, 1976] Allen L. Edwards. *An Introduction to Linear Regression and Correlation*. W. H. Freema, San Francisco, Cal., 1976.

[Goodman and Kruskal, 1972] Leo A. Goodman and W. H. Kruskal. Measures of association for cross-classification, iv. *Journal of the American Statistical Association*, 67:415–421, 1972.

[Joshi and Schabes, 1991] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars and lexicalized grammars. In *Maurice Nivat and Andreas Podelski, editors*, Definability and Recognizability of Sets of Trees. *Elsevier*, 1991.

[Kleinberg and Tardos, 2000] Jon M. Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM*, 49(5):616–639, 2000.

[Kohavi and John, 1997] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence Journal*, 97:273–324, 1997.

[Marciniak and Strube, 2004] Tomasz Marciniak and Michael Strube. Classification-based generation using TAG. In *Proceedings of the 3rd International Conference on Natural Language Generation*, Brockenhurst, UK, 14-16 July, 2004, pages 100–109, 2004.

[Marciniak and Strube, 2005] 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.

[Mellish and Evans, 2004] Chris Mellish and Roger Evans. Implementation architectures for natural language generation. *Natural Language Engineering*, 10(3/4):261–282, 2004.

[Mellish *et al.*, 2004] Chris Mellish, Mike Reape, Donia Scott, Lynne Cahill, Roger Evans, and Daniel Paiva. A reference architecture for generation systems. *Natural Language Engineering*, 10(3/4):227–260, 2004.

[Meteer, 1990] Marie W. Meteer. Abstract linguistic resources for text planning. In *Proceedings of the 5th International Workshop on Natural Language Generation*, Pittsburgh, PA, 3-6 June 1990, pages 62–69, 1990.

[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, 2004] Ehud Reiter and Somayajulu Sripada. Contextual influences on near-synonym choice. In *Proceedings of the 3rd International Conference on Natural Language Generation*, Brockenhurst, UK, 14-16 July, 2004, pages 161–170, 2004.

[Reiter, 1994] Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proc. of the 7th International Workshop on Natural Language Generation*, Kennebunkport, MA, 21-24 June 1994, pages 160–173, 1994.

[Roth and Yih, 2004] Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, Boston, Mass., May 2-7, 2004, pages 1–8, 2004.

[Webber and Joshi, 1998] Bonnie Lynn Webber and Aravind Joshi. Anchoring a lexicalized tree-adjoining 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.

[Witten and Frank, 2000] Ian H. Witten and Eibe Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, Cal., 2000.