Combining Referring Expression Generation and Surface Realization: A Corpus-Based Investigation of Architectures

Sina Zarrieß **Jonas Kuhn**
Institut für maschinelle Sprachverarbeitung
University of Stuttgart, Germany
sina.zarriess,jonas.kuhn@ims.uni-stuttgart.de

Abstract

We suggest a generation task that integrates discourse-level referring expression generation and sentence-level surface realization. We present a data set of German articles annotated with deep syntax and referents, including some types of implicit referents. Our experiments compare several architectures varying the order of a set of trainable modules. The results suggest that a revision-based pipeline, with intermediate linearization, significantly outperforms standard pipelines or a parallel architecture.

1 Introduction

Generating well-formed linguistic utterances from an abstract non-linguistic input involves making a multitude of conceptual, discourse-level as well as sentence-level, lexical and syntactic decisions. Work on rule-based natural language generation (NLG) has explored a number of ways to combine these decisions in an architecture, ranging from integrated systems where all decisions happen jointly (Appelt, 1982) to strictly sequential pipelines (Reiter and Dale, 1997). While integrated or interactive systems typically face issues with efficiency and scalability, they can directly account for interactions between discourse-level planning and linguistic realization. For instance, Rubinoff (1992) mentions Example (1) where the sentence planning component needs to have access to the lexical knowledge that "order" and not "home" can be realized as a verb in English.

(1) a. *John homed him with an order.
b. John ordered him home.

In recent data-driven generation research, the focus has somewhat shifted from full data-to-text systems to approaches that isolate well-defined subproblems from the NLG pipeline. In particular, the tasks of surface realization and referring expression generation (REG) have received increasing attention using a number of available annotated data sets (Belz and Kow, 2010; Belz et al., 2011). While these single-task approaches have given rise to many insights about algorithms and corpus-based modelling for specific phenomena, they can hardly deal with aspects of the architecture and interaction between generation levels.

This paper suggests a middle ground between full data-to-text and single-task generation, combining two well-studied NLG problems. We integrate a discourse-level approach to REG with sentence-level surface realization in a data-driven framework. We address this integrated task with a set of components that can be trained on flexible inputs which allows us to systematically explore different ways of arranging the components in a generation architecture. Our main goal is to investigate how different architectural set-ups account for interactions between generation decisions at the level of referring expressions (REs), syntax and word order.

Our basic set-up is inspired from the Generating Referring Expressions in Context (GREC) tasks, where candidate REs have to be assigned to instances of a referent in a Wikipedia article (Belz and Kow, 2010). We have created a dataset of German texts with annotations that extend this standard in three substantial ways: (i) our domain consists of articles about *robbery* events that mainly involve two main referents, a *victim* and a *perpetrator* (*perp*), (ii) annotations include deep and shallow syntactic relations similar to the representations used in (Belz et al., 2011) (iii) annotations include empty referents, as e.g. in passives and nominalizations directing attention to the phenomenon of implicit reference, which is largely understudied in NLG. Figure 1 presents an example for a deep syntax tree with underspecified RE

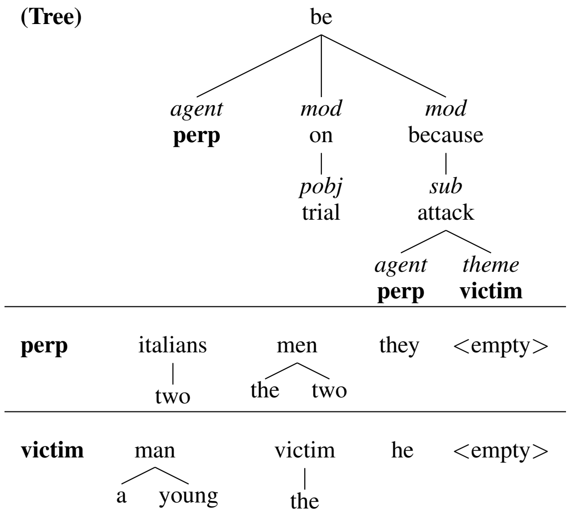

Figure 1: Underspecified tree with RE candidates

slots and lists of candidates REs for each referent.

Applying a strictly sequential pipeline on our data, we observe incoherent system output that is related to an interaction of generation levels, very similar to the interleaving between sentence planning and lexicalization in Example (1). A pipeline that first inserts REs into the underspecified tree in Figure 1, then generates syntax and finally linearizes, produces inappropriate sentences like (2-a).

(2) a. *[The two men]$_p$ are on trial because of an attack by [two italians]$_p$ on [a young man]$_v$.
b. [Two italians]$_p$ are on trial because of an attack on [a young man]$_v$.

Sentence (2-a) is incoherent because the syntactic surface obscurs the intended meaning that "two italians" and "the two men" refer to the same referent. In order to generate the natural Sentence (2-b), the RE component needs information about linear precedence of the two *perp* instances and the nominalization of "attack". These types of interactions between referential and syntactic realization have been thoroughly discussed in theoretical accounts of textual coherence, as e.g. Centering Theory (Grosz et al., 1995).

The integrated modelling of REG and surface realization leads to a considerable expansion of the choice space. In a sentence with 3 referents that each have 10 RE candidates and can be freely ordered, the number of surface realizations increases from 6 to $6 \cdot 10^3$, assuming that the remaining words can not be syntactically varied. Thus, even when the generation problem is restricted to these tasks, a fully integrated architecture faces scalability issues on realistic corpus data.

In this work, we assume a modular set-up of the generation system that allows for a flexible ordering of the single components. Our experiments vary 3 parameters of the generation architecture: 1) the sequential order of the modules, 2) parallelization of modules, 3) joint vs. separate modelling of implicit referents. Our results suggest that the interactions between RE and syntax can be modelled in sequential generation architecture where the RE component has access to information about syntactic realization and an approximative, intermediate linearization. Such a system is reminiscent of earlier work in rule-based generation that implements an interactive or revision-based feedback between discourse-level planning and linguistic realisation (Hovy, 1988; Robin, 1993).

2 Related Work

Despite the common view of NLG as a pipeline process, it is a well-known problem that high-level, conceptual knowledge and low-level linguistic knowledge are tightly interleaved (Danlos, 1984; Mellish et al., 2000). In rule-based, strictly sequential generators these interactions can lead to a so-called *generation gap*, where a downstream module cannot realize a text or sentence plan generated by the preceding modules (Meteer, 1991; Wanner, 1994). For this reason, a number of other architectures has been proposed, see De Smedt et al. (1996) for an overview. For reasons of tractability and scalability, many practical NLG systems still have been designed as sequential pipelines that follow the basic layout of macroplanning-microplanning-linguistic realization (Reiter, 1994; Cahill et al., 1999; Bateman and Zock, 2003).

In recent data-driven research on NLG, many single tasks have been addressed with corpus-based methods. For surface realization, the standard set-up is to regenerate from syntactic representations that have been produced for realistic corpus sentences. The first widely known statistical approach by Langkilde and Knight (1998) used language-model n-gram statistics on a word lattice of candidate realisations to guide a ranker. Subsequent work explored ways of exploiting linguistically annotated data for trainable generation models (Ratnaparkhi, 2000; Belz, 2005). Work on data-driven approaches has led to insights about the importance of linguistic features for sentence linearization decisions (Ringger et al., 2004; Filippova and Strube, 2007; Cahill and Riester, 2009). (Zarrieß et al., 2012) have recently argued that the good performance of these linguistically motivated word order models, which exploit morpho-syntactic features of noun phrases (i.e. referents), is related to the fact that these morpho-syntactic features implicitly encode a lot of knowledge about the underlying discourse or information structure.

A considerable body of REG research has been done in the paradigm established by Dale (1989; 1995). More closely related to our work are approaches in the line of Siddarthan and Copestake (2004) or Belz and Varges (2007) who generate contextually appropriate REs for instances of a referent in a text. Belz and Varges (2007)'s GREC data set includes annotations of implicit subjects in coordinations. Zarrieß et al. (2011) deal with implicit subjects in passives, proposing a set of heuristics for adding these agents to the generation input. Roth and Frank (2012) acquire automatic annotations of implicit roles for the purpose of studying coherence patterns in texts. Implicit referents have also received attention for the analysis of semantic roles (Gerber and Chai, 2010; Ruppenhofer et al., 2010).

Statistical methods for data-to-text generation have been explored only recently. Belz (2008) trains a probabilistic CFG to generate weather forecasts, Chen et al. (2010) induce a synchronous grammar to generate sportcaster text. Both address a restricted domain where a direct alignment between units in the non-linguistic representation and the linguistic utterance can be learned. Marciniak and Strube (2005) propose an ILP model for global optimization in a generation task that is decomposed into a set of classifiers. Bohnet et al. (2011) deal with multi-level generation in a statistical framework and in a less restricted domain. They adopt a standard sequential pipeline approach.

Recent corpus-based generation approaches faced the problem that existing standard treebank representations for parsing or other analysis tasks do not necessarily fit the needs of generation (Bohnet et al., 2010; Wanner et al., 2012). Zarrieß et al. (2011) discuss the problem of an input representation that is appropriately underspecified for the realistic generation of voice alternations.

3 The Data Set

The data set for our generation experiments consists of 200 newspaper articles about robbery events. The articles were extracted from a large German newspaper corpus. A complete example text with RE annotations is given in Figure 2, Table 1 summarizes some data set statistics.

3.1 RE annotation

The RE annotations mark explicit and implicit mentions of referents involved in the robbery event described in an article. Explicit mentions are marked as spans on the surface sentence, labeled with the referent's role and an ID. We annotate the following referential roles: (i) *perpetrator* (*perp*), (ii) *victim*, (iii) *source*, according to the core roles of the *Robbery* frame in English FrameNet. We include *source* since some texts do not mention a particular *victim*, but rather the location of the robbery (e.g. a bank, a service station). The ID distinguishes referents that have the same role, e.g. "the husband" and the "young family" in Sentences (3-a) and (3-d) in Figure 2. Each RE is linked to its syntactic head. This complies with the GREC data sets, and is also useful for further annotation of the deep syntax level (see Section 3.2).

The RE implicit mentions of *victim*, *perp*, and *source* are annotated as attributes of their syntactic heads in the surface sentence. We consider the following types of implicit referents: (i) agents in passives (e.g. "robbed" in (3-a)), (ii) arguments of nominalizations (e.g. "resistance" in (3-e)), (iii) possessives (e.g. "watch" in (3-f)), (iv) missing subjects in coordinations. (e.g. "flee" in (3-f))

The brat tool (Stenetorp et al., 2012) was used for annotation. We had 2 annotators with a computational linguistic background, provided with annotation guidelines. They were trained on a set of 20 texts. We measure a good agreement on another set of 15 texts: the simple pairwise agreement for explicit mentions is 95.14%-96.53% and 78.94%-76.92% for implicit mentions.[1]

3.2 Syntax annotation

The syntactic annotation of our data includes two layers: shallow and deep, labeled dependencies, similar to the representation used in surface realization shared tasks (Belz et al., 2011). We use

[1]Standard measures for the "above chance annotator agreement" are only defined for task where the set of annotated items is pre-defined.

(3) a. Junge Familie$_{v:0}$ auf dem Heimweg$_{poss:v}$ ausgeraubt$_{ag:p}$
Young family on the way home$_{poss:v}$ robbed$_{ag:p}$

b. Die Polizei sucht nach zwei ungepflegt wirkenden jungen Männern im Alter von etwa 25 Jahren$_{p:0}$.
The police looks for two shabby-looking young men of about 25 years.

c. Sie$_{p:0}$ sollen am Montag gegen 20 Uhr eine junge Familie mit ihrem sieben Monate alten Baby$_{v:0}$ auf dem Heimweg$_{poss:v}$ von einem Einkaufsbummel überfallen und ausgeraubt haben.
They are said to on Monday around 20 o'clock a young family with their seven month old baby on the way home$_{poss:v}$ from a shopping tour attacked and robbed have.

d. Wie die Polizei berichtet, drohten die zwei Männer$_{p:0}$ dem Ehemann$_{v:1}$, ihn$_{v:1}$ zusammenzuschlagen.
As the police reports, threatened the two men the husband him beat up.

e. Er$_{v:1}$ gab deshalb seine$_{v:1}$ Brieftasche ohne Gegenwehr$_{ag:v,the:p}$ heraus.
He gave therefore his wallet without resistance$_{ag:v,the:p}$ out.

f. Anschließend nahmen ihm$_{v:1}$ die Räuber$_{p:0}$ noch die Armbanduhr$_{poss:v}$ ab und flüchteten$_{ag:p}$.
Afterwards took him the robbers also the watch$_{poss:v}$ off and fleed$_{ag:p}$.

Figure 2: Example text with RE annotations, oval boxes mark *victim* mentions, square boxes mark *perp* mentions, heads of implicit arguments are underlined

the Bohnet (2010) dependency parser to obtain an automatic annotation of shallow or surface dependencies for the corpus sentences.

The deep syntactic dependencies are derived from the shallow layer by a set of hand-written transformation rules. The goal is to link referents to their main predicate in a uniform way, independently of the surface-syntactic realization of the verb. We address passives, nominalizations and possessives corresponding to the contexts where we annotated implicit referents (see above). The transformations are defined as follows:

1. remove auxiliary nodes, verb morphology and finiteness, a tense feature distinguishes past and present, e.g. "haben:AUX überfallen:VVINF" (*have attacked*) maps to "überfallen:VV:PAST" (*attack:PAST*)
2. map subjects in actives and oblique agents in passives to "agents"; objects in actives and subjects in passive to "themes", e.g. *victim/subj was attacked by perp/obl-ag* maps to *perp/agent attack victim/theme*
3. attach particles to verb lemma, e.g. "gab" ... "heraus" in (3-e) is mapped to "herausgeben" (*give to*)
4. map nominalized to verbal lemmas, their prepositional and genitive arguments to semantic subjects and objects, e.g. *attack on victim* is mapped to *attack victim/theme*
5. normalize prenominal and genitive postnominal posessives, e.g. "seine Brieftasche" (*his wallet*) and "die Brieftasche des Opfers" (*the wallet of the victim*) map to "die Brieftasche POSS victim" (*the wallet of victim*), only applies if possessive is an annotated RE

Nominalizations are mapped to their verbal base forms on the basis of lexicalized rules for the nominalized lemmas observed in the corpus. The other transformations are defined on the shallow dependency annotation.

# sentences	2030
# explicit REs	3208
# implicit REs	1778
# passives	383
# nominalizations	393
# possessives	1150

Table 1: Basic annotation statistics

3.3 Multi-level Representation

In the final representation of our data set, we integrate the RE and deep syntax annotation by replacing subtrees corresponding to an RE span. The RE slot in the tree of the sentence is labeled with its referential role and its ID. All RE subtrees for a referent in a text are collected in a candidate list which is initialized with three default REs: (i) a pronoun, (ii) a default nominal (e.g. "the victim"), (iii) the empty RE. In contrast to the GREC data sets, our RE candidates are not represented as the original surface strings, but as non-linearized subtrees. The resulting multi-layer representation for each text is structured as follows:

1. unordered deep trees with RE slots ($deepSyn_{-re}$)
2. unorderd shallow trees with RE slots ($shallowSyn_{-re}$)
3. unordered RE subtrees
4. linearized, fully specified surface trees ($linSyn_{+re}$)
5. alignments between nodes in 1., 2., 4.

The generation components in Section 4 also use intermediate layers where REs are inserted into the deep trees ($deepSyn_{+re}$) or shallow trees ($shallowSyn_{+re}$).

Nodes in unordered trees are deterministically sorted by their : 1. distance to the root, 2. label,

3. PoS tag, 4. lemma. The generation components traverse the nodes in this the order.

4 Generation Systems

Our main goal is to investigate different architectures for combined surface realization and referring expression generation. We assume that this task is split into three main modules: a syntax generator, an REG component, and a linearizer. The components are implemented in a way that they can be trained and applied on varying inputs, depending on the pipeline. Section 4.1 describes the basic set-up of our components. Section 4.2 defines the architectures that we will compare in our experiments (Section 5). Section 4.3 presents the implementation of the underlying feature models.

4.1 Components

4.1.1 SYN: Deep to Shallow Syntax

For mapping deep to shallow dependency trees, the syntax generator induces a probabilistic tree transformation. The transformations are restricted to verb nodes in the deep tree (possessives are handled in the RE module) and extracted from the alignments between the deep and shallow layer in the training input. As an example, the deep node "attack:VV" aligns to "have:AUX attacked:VVINF", "attacks:VVFIN", "the:ART attack:NN on:PRP". The learner is implemented as a ranking component, trained with SVMrank (Joachims, 2006). During training, each instance of a verb node has one optimal shallow dependency alignment and a set of distractor candidates. During testing, the module has to pick the best shallow candidate according to its feature model.

In our crossvalidation set-up (see Section 5), we extract, on average, 374 transformations from the training sets. This set subdivides into non-lexicalized and lexicalized transformations. The mapping rule in (4-a) that simply rewrites the verb underspecified PoS tag to the finite verb tag in the shallow tree illustrates the non-lexicalized case. Most transformation rules (335 out of 374 on average) are lexicalized for a specific verb lemma and mostly transform nominalizations as in rule (4-b) and particles (see Section 3.2).

(4) a. (x,`lemma`,VV,y) → (x,`lemma`,VVFIN,y)
b. (x,überfallen/*attack*,VV,y) → (x,bei/*at*,PREP,y), (z,Überfall/*attack*,NN,x),(q,der/*the*,ART,z)

The baseline for the verb transformation component is a two-step procedure: 1) pick a lexicalized rule if available for that verb lemma, 2) pick the most frequent transformation.

4.1.2 REG: Realizing Referring Expressions

Similar to the syntax component, the REG module is implemented as a ranker that selects surface RE subtrees for a given referential slot in a deep or shallow dependency tree. The candidates for the ranking correspond to the entire set of REs used for that referential role in the original text (see Section 3.1). The basic RE module is a joint model of all RE types, i.e. nominal, pronominal and empty realizations of the referent. For the experiment in Section 5.4, we use an additional separate classifier for implicit referents, also trained with SVMrank. It uses the same feature model as the full ranking component, but learns a binary distinction for implicit or explicit mentions of a referent. The explicit mentions will be passed to the RE ranking component.

The baseline for the REG component is defined as follows: if the preceding and the current RE slot are instances of the same referent, realize a pronoun, else realize the longest nominal RE candidate that has not been used in the preceding text.

4.1.3 LIN: Linearization

For linearization, we use the state-of-the-art dependency linearizer described in Bohnet et al. (2012). We train the linearizer on an automatically parsed version of the German TIGER treebank (Brants et al., 2002). This version was produced with the dependency parser by Bohnet (2010), trained on the dependency conversion of TIGER by Seeker and Kuhn (2012).

4.2 Architectures

Depending on the way the generation components are combined in an architecture, they will have access to different layers of the input representation. The following definitions of architectures recur to the layers introduced in Section 3.3.

4.2.1 First Pipeline

The first pipeline corresponds most closely to a standard generation pipeline in the sense of (Reiter and Dale, 1997). REG is carried out prior to surface realization such that the RE component does not have access to surface syntax or word order whereas the SYN component has access to fully specified RE slots.

- training

1. train REG: $(deepSyn_{-re}, deepSyn_{+re})$
2. train SYN: $(deepSyn_{+re}, shallowSyn_{+re})$

- prediction
  1. apply REG: $deepSyn_{-re} \rightarrow deepSyn_{+re}$
  2. apply SYN: $deepSyn_{+re} \rightarrow shallowSyn_{+re}$
  3. linearize: $shallowSyn_{+re} \rightarrow linSyn_{+re}$

4.2.2 Second Pipeline

In the second pipeline, the order of the RE and SYN component is switched. In this case, REG has access to surface syntax without word order but the surface realization is trained and applied on trees with underspecified RE slots.

- training
  1. train SYN: $(deepSyn_{-re}, shallowSyn_{-re})$
  2. train REG: $(shallowSyn_{-re}, shallowSyn_{+re})$
- prediction
  1. apply SYN: $deepSyn_{-re} \rightarrow shallowSyn_{-re}$
  2. apply REG: $shallowSyn_{-re} \rightarrow shallowSyn_{+re}$
  3. linearize: $shallowSyn_{+re} \rightarrow linSyn_{+re}$

4.2.3 Parallel System

A well-known problem with pipeline architectures is the effect of error propagation. In our parallel system, the components are trained independently of each other and applied in parallel on the deep syntactic input with underspecified REs.

- training
  1. train SYN: $(deepSyn_{-re}, shallowSyn_{-re})$
  2. train REG: $(deepSyn_{-re}, deepSyn_{+re})$
- prediction
  1. apply REG and SYN: $deepSyn_{-re} \rightarrow shallowSyn_{+re}$
  2. linearize: $shallowSyn_{+re} \rightarrow linSyn_{+re}$

4.2.4 Revision-based System

In the revision-based system, the RE component has access to surface syntax and a preliminary linearization, called *prelinSyn*. In this set-up, we apply the linearizer first on trees with underspecified RE slots. For this step, we insert the default REs for the referent into the respective slots. After REG, the tree is linearized once again.

- training
  1. train SYN on gold pairs of $(deepSyn_{-re}, shallowSyn_{-re})$
  2. train REG on gold pairs of $(prelinSyn_{-re}, prelinSyn_{+re})$
- prediction
  1. apply SYN: $deepSyn_{-re} \rightarrow shallowSyn_{-re}$
  2. linearize: $shallowSyn_{-re} \rightarrow prelinSyn_{-re}$
  3. apply REG: $prelinSyn_{-re} \rightarrow prelinSyn_{+re}$
  4. linearize: $prelinSyn_{+re} \rightarrow linSyn_{+re}$

4.3 Feature Models

The implementation of the feature models is based on a general set of templates for the SYN and REG component. The exact form of the models depends on the input layer of a component in a given architecture. For instance, when SYN is trained on $deepSyn_{-re}$, the properties of the children nodes are less specific for verbs that have RE slots as their dependents. When the SYN component is trained on $deepSyn_{+re}$, lemma and POS of the children nodes are always specified.

The feature templates for SYN combine properties of the shallow candidate nodes (label, PoS and lemma for top node and its children) with the properties of the instance in the tree: (i) lemma, tense, (ii) sentence is a header, (iii) label, PoS, lemma of mother node, children and grandchildren nodes (iv) number, lemmas of other verbs in the sentence.

The feature templates for REG combine properties of the candidate RE (PoS and lemma for top node and its children, length) with properties of the RE slot in the tree: lemma, PoS and labels for the (i) mother node, (ii) grandmother node, (iii) uncle and sibling nodes. Additionally, we implement a small set of global properties of a referent in a text: (i) identity is known, (ii) plural or singular referent, (iii) age is known, and a number of contextual properties capturing the previous referents and their predicted REs: (i) role and realization of the preceding referent, (ii) last mention of the current referent, (iii) realization of the referent in the header.

5 Experiments

In this experimental section, we provide a corpus-based evaluation of the generation components and architectures introduced in Section 4. In the following, Section 5.1 presents the details of our evaluation methodology. In Section 5.2, we discuss the first experiment that evaluates the pipeline architectures and the single components on oracle inputs. Section 5.3 describes an experiment which compares the parallel and the revision-based architecture against the pipeline. In Section 5.4, we compare two methods for dealing with the implicit referents in our data. Section 5.5 provides some general discussion of the results.

		Sentence overlap			SYN Accuracy		RE Accuracy		
Input	System	BLEU	NIST	BLEU$_r$	String	Type	String	Type	Impl
$deepSyn_{-re}$	Baseline	42.38	9.9	47.94	35.66	44.81	33.3	36.03	50.43
$deepSyn_{-re}$	1st pipeline	54.65	11.30	59.95	57.09	68.15	54.61	71.51	84.72
$deepSyn_{-re}$	2nd pipeline	54.28	11.25	59.62	59.14	68.58	52.24	68.2	82
gold $deepSyn_{+re}$	SYN→LIN	63.9	12.7	62.86	60.83	69.74	100	100	100
gold $shallowSyn_{-re}$	REG→LIN	60.57	11.87	68.06	100	100	60.53	75.86	88.86
gold $shallowSyn_{+re}$	LIN	79.17	13.91	72.7	100	100	100	100	100

Table 2: Evaluating pipeline architectures against the baseline and upper bounds

5.1 Evaluation Measures

We split our data set into 10 splits of 20 articles. We use one split as the development set, and cross-validate on the remaining splits. In each case, the downstream modules of the pipeline will be trained on the jackknifed training set.

Text normalization: We carry out automatic evaluation calculated on lemmatized text without punctuation, excluding additional effects that would be introduced from a morphology generation component.

Measures: First, we use a number of evaluation measures familiar from previous generation shared tasks:

1. BLEU, sentence-level geometric mean of 1- to 4-gram precision, as in (Belz et al., 2011)
2. NIST, sentence-level n-gram overlap weighted in favour of less frequent n-grams, as in (Belz et al., 2011)
3. RE Accuracy on String, proportion of REs selected by the system with a string identical to the RE string in the original corpus, as in (Belz and Kow, 2010)
4. RE Accuracy on Type, proportion of REs selected by the system with an RE type identical to the RE type in the original corpus, as in (Belz and Kow, 2010)

Second, we define a number of measures motivated by our specific set-up of the task:

1. BLEU$_r$, sentence-level BLEU computed on post-processed output where predicted referring expressions for *victim* and *perp* are replaced in the sentences (both gold and predicted) by their original role label, this score does not penalize lexical mismatches between corpus and system REs
2. RE Accuracy on Impl, proportion of REs predicted correctly as implicit/non-implicit
3. SYN Accuracy on String, proportion of shallow verb candidates selected by the system with a string identical to the verb string in the original corpus
4. SYN Accuracy on Type, proportion of shallow verb candidates selected by the system with a syntactic category identical to the category in the original corpus

5.2 Pipelines and Upper Bounds

The first experiment addresses the first and second pipeline introduced in Section 4.2.1 and 4.2.2. The baseline combines the baseline version of the SYN component (Section 4.1.1) and the REG component (Section 4.1.2) respectively. As we report in Table 2, both pipelines largely outperform the baseline. Otherwise, they obtain very similar scores in all measures with a small, weakly significant tendency for the first pipeline. The only remarkable difference is that the accuracy of the individual components is, in each case, lower when they are applied as the second step in the pipeline. Thus, the RE accuracy suffers from mistakes from the predicted syntax in the same way that the quality of syntax suffers from predicted REs.

The three bottom rows in Table 2 report the performance of the individual components and linearization when they are applied to inputs with an REG and SYN oracle, providing upper bounds for the pipelines applied on $deepSyn_{-re}$. When REG and linearization are applied on $shallowSyn_{-re}$ with gold shallow trees, the BLEU score is lower (60.57) as compared to the system that applies syntax and linearization on $deepSyn_{+re}$, deep trees with gold REs (BLEU score of 63.9). However, the BLEU$_r$ score, which generalizes over lexical RE mismatches, is higher for the REG→LIN components than for SYN→LIN. Moreover, the BLEU$_r$ score for the REG→LIN system comes close to the upper bound that applies linearization on $linSyn_{+re}$, gold shallow trees with gold REs (BLEU$_r$ of 72.4), whereas the difference in standard BLEU and NIST is high. This effect indicates that the RE prediction mostly decreases BLEU due to lexical mismatches, whereas the syntax prediction is more likely to have a negative impact on final linearization.

The error propagation effects that we find in the first and second pipeline architecture clearly show that decisions at the levels of syntax, reference and word order interact, otherwise their predic-

Input	System	BLEU	NIST	BLEU$_r$
$deepSyn_{-re}$	1st pipeline	54.65	11.30	59.95
$deepSyn_{-re}$	Parallel	54.78	11.30	60.05
$deepSyn_{-re}$	Revision	56.31	11.42	61.30

Table 3: Architecture evaluation

tion would not affect each other. In particular, the REG module seems to be affected more seriously, the String Accuracy decreases from 60.53 on gold shallow trees to 52.24 on predicted shallow trees whereas the Verb String Accuracy decreases from 60.83 on gold REs to 57.04 on predicted REs.

5.3 Revision or parallelism?

The second experiment compares the first pipeline against the parallel and the revision-based architecture introduced in Section 4.2.3 and 4.2.4. The evaluation in Table 3 shows that the parallel architecture improves only marginally over the pipeline. By contrast, we obtain a clearly significant improvement for the revision-based architecture on all measures. The fact that this architecture significantly improves the BLEU, NIST and the BLEU$_r$ score of the parallel system indicates that the REG benefits from the predicted syntax when it is approximatively linearized. The fact that also the BLEU$_r$ score improves shows that a higher lexical quality of the REs leads to better final linearizations.

Table 4 shows the performance of the REG module on varying input layers, providing a more detailed analysis of the interaction between RE, syntax and word order. In order to produce the $deeplinSyn_{-re}$ layer, deep syntax trees with approximative linearizations, we preprocessed the deep trees by inserting a default surface transformation for the verb nodes. We compare this input for REG against the $prelinSyn_{-re}$ layer used in the revision-based architecture, and the $deepSyn_{-re}$ layer used in the pipeline and the parallel architecture. The REG module benefits from the linearization in the case of $deeplinSyn_{-re}$ and $prelinSyn_{-re}$, outperforming the component trained applied on the non-linearized deep syntax trees. However, the REG module applied on $prelinSyn_{-re}$, predicted shallow and linearized trees, clearly outperforms the module applied on $deeplinSyn_{-re}$. This shows that the RE prediction can actually benefit from the predicted shallow syntax, but only when the predicted trees are approximatively linearized. As an upper bound, we report the performance obtained on

		RE Accuracy		
Input	System	String	Type	Impl
$deepSyn_{-re}$	RE	54.61	71.51	84.72
$deeplinSyn_{-re}$	RE	56.78	72.23	84.71
$prelinSyn_{-re}$	RE	58.81	74.34	86.37
gold $linSyn_{-re}$	RE	68.63	83.63	94.74

Table 4: RE generation from different input layers

$linSyn_{-re}$, gold shallow trees with gold linearizations. This set-up corresponds to the GREC tasks. The gold syntax leads to a huge increase in performance.

These results strengthen the evidence from the previous experiment that decisions at the level of syntax, reference and word order are interleaved. A parallel architecture that simply "circumvents" error propagation effects by making decisions independent of each other is not optimal. Instead, the automatic prediction of shallow syntax can positively impact on RE generation if these shallow trees are additionally processed with an approximative linearization step.

5.4 A joint treatment of implicit referents?

The previous experiments have pursued a joint approach for modeling implicit referents. The hypothesis for this experiment is that the SYN component and the intermediate linearization in a revision-based architecture could benefit from a separate treatment of implicit referents since verb alternations like passive or nominalization often involve referent deletions.

The evaluation in Table 5 provides contradictory results depending on the evaluation measure. For the first pipeline, the system with a separate treatment of implicit referents significantly outperforms the joint system in terms of BLEU. However, the BLEU$_r$ score does not improve. In the revision-based architecture, we do not find a clear result for or against a joint modelling approach. The revision-based system with disjoint modelling of implicits shows a slight, non-significant increase in BLEU score. By contrast, the BLEU$_r$ score is signficantly better for the joint approach. We experimented with parallelization of syntax generation and prediction of implicit referents in a revision-based system. This has a small positive effect on the BLEU$_r$ score and a small negative effect on the plain BLEU and NIST score. These contradictory scores might indicate that the automatic evaluation measures cannot capture all aspects of text quality, an issue that we discuss in the following.

(5) Generated by sequential system:

a.

Deshalb	gab	dem Täter	seine	Brieftasche	ohne	daß	das Opfer	Widerstand	leistet	heraus.
Therefore	gave	to the robber	his	wallet	without	that	the victim	resistance	shows	out.

b.

Er	nahm	anschließend	dem Opfer	die Armbanduhr	ab	und	der Täter	flüchtete.
He	takes	afterwards	the victim	the watch	off	and	the robber	fleed.

(6) Generated by revision-based system:

a.

Das Opfer	gibt	deshalb	seine	Brieftasche	ohne	Widerstand	zu	leisten	heraus.
The victim	gave	therefore	his	wallet	without	resistance	to	show	out.

b.

Anschließend	nahm	der Täter	dem Opfer	die Armbanduhr	ab	und	flüchtete.
Afterwards	took	the robber	the victim	the watch	off	and	fleed.

Figure 3: Two automatically generated outputs for the Sentences (3e-f) in Figure 2.

Joint	System	BLEU	NIST	BLEU$_r$
+	1st pipeline	54.65	11.30	59.95
-	1st pipeline	55.38	11.48	59.52
+	Revision	56.31	11.42	61.30
-	Revision	56.42	11.54	60.52
-	Parallel+Revision	56.29	11.51	60.63

Table 5: Implicit reference and architectures

5.5 Discussion

The results presented in the preceding evaluations consistenly show the tight connections between decisions at the level of reference, syntax and word order. These interactions entail highly interdependent modelling steps: Although there is a direct error propagation effect from predicted verb transformation on RE accuracy, predicted syntax still leads to informative intermediate linearizations that improve the RE prediction. Our optimal generation architecture thus has a sequential set-up, where the first linearization step can be seen as an intermediate feedback that is revised in the final linearization. This connects to work in, e.g. (Hovy, 1988; Robin, 1993).

In Figure 3, we compare two system outputs for the last two sentences of the text in Figure 2. The output of the sequential system is severely incoherent and would probably be rejected by a human reader: In sentence (5a) the *victim* subject of an active verb is deleted, and the relation between the possessive and the embedded *victim* RE is not clear. In sentence (5b) the first conjunct realizes a pronominal *perp* RE and the second conjunct a nominal *perp* RE. The output of the revision-based system reads much more natural. This example shows that the extension of the REG problem to texts with more than one main referent (as in the GREC data set) yields interesting inter-sentential interactions that affect textual coherence.

We are aware of the fact that our automatic evaluation might only partially render certain effects, especially with respect to textual coherence. It is likely that the BLEU scores do not capture the magnitude of the differences in text quality illustrated by the Examples (5-6). Ultimately, a human evaluation for this task is highly desirable. We leave this for future work since our integrated set-up rises a number of questions with respect to evaluation design. In a preliminary analysis, we noticed the problem that human readers find it difficult to judge discourse-level properties of a text like coherence or naturalness when the generation output is not perfectly grammatical or fluent at the sentence level.

6 Conclusion

We have presented a data-driven approach for investigating generation architectures that address discourse-level reference and sentence-level syntax and word order. The data set we created for our experiments basically integrates standards from previous research on REG and surface realization and extends the annotations to further types of implicit referents. Our results show that interactions between the different generation levels are best captured in a sequential, revision-based pipeline where the REG component has access to predictions from the syntax and the linearization module. These empirical findings obtained from experiments with generation architectures have clear connections to theoretical accounts of textual coherence.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) in SFB 732 *Incremental Specification in Context*, project D2.

References

Douglas Edmund Appelt. 1982. *Planning natural language utterances to satisfy multiple goals*. Ph.D. thesis, Stanford, CA, USA.

John Bateman and Michael Zock. 2003. Natural Language Generation. In Ruslan Mitkov, editor, *The Oxford Handbook of Computational Linguistics*. Oxford University Press.

Anja Belz and Eric Kow. 2010. The GREC Challenges 2010: overview and evaluation results. In *Proc. of the 6th International Natural Language Generation Conference*, INLG '10, pages 219–229, Stroudsburg, PA, USA.

Anja Belz and Sebastian Varges. 2007. Generation of repeated references to discourse entities. In *Proc. of the 11th European Workshop on Natural Language Generation*, ENLG '07, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proc. of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France, September. Association for Computational Linguistics.

Anja Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proc. of the 10th European Workshop on Natural Language Generation*, pages 15–23.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.*, 14(4):431–455, October.

Bernd Bohnet, Leo Wanner, Simon Milles, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proc. of the 23rd International Conference on Computational Linguistics*, Beijing, China.

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. <stumaba >: From deep representation to surface. In *Proc. of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 232–235, Nancy, France, September.

Bernd Bohnet, Anders Björkelund, Jonas Kuhn, Wolfgang Seeker, and Sina Zarriess. 2012. Generating non-projective word order in statistical linearization. In *Proc. of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 928–939, Jeju Island, Korea, July.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proc. of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China, August.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proc. of the Workshop on Treebanks and Linguistic Theories*.

Aoife Cahill and Arndt Riester. 2009. Incorporating Information Status into Generation Ranking. In *Proc. of the 47th Annual Meeting of the ACL*, pages 817–825, Suntec, Singapore, August.

Lynne Cahill, Christy Doran, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, Donia Scott, and Neil Tipper. 1999. In search of a reference architecture for nlg systems. In *Proc. of the European Workshop on Natural Language Generation (EWNLG)*, pages 77–85.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37:397–435.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Robert Dale. 1989. Cooking up referring expressions. In *Proc. of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 68–75, Vancouver, British Columbia, Canada, June.

Laurence Danlos. 1984. Conceptual and linguistic decisions in generation. In *Proc. of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pages 501–504, Stanford, California, USA, July.

Koenraad De Smedt, Helmut Horacek, and Michael Zock. 1996. Architectures for natural language generation: Problems and perspectives. In *Trends In Natural Language Generation: An Artifical Intelligence Perspective*, pages 17–46. Springer-Verlag.

Katja Filippova and Michael Strube. 2007. Generating constituent order in german clauses. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Matthew Gerber and Joyce Chai. 2010. Beyond nombank: A study of implicit arguments for nominal predicates. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592, Uppsala, Sweden, July.

Barbara J. Grosz, Aravind Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Eduard H. Hovy. 1988. Planning coherent multisentential text. In *Proc. of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 163–169, Buffalo, New York, USA, June.

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proc. of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Tomasz Marciniak and Michael Strube. 2005. Beyond the pipeline: discrete optimization in nlp. In *Proc. of the 9th Conference on Computational Natural Language Learning*, CONLL '05, pages 136–143, Stroudsburg, PA, USA.

Chris Mellish, Roger Evans, Lynne Cahill, Christy Doran, Daniel Paiva, Mike Reape, Donia Scott, and Neil Tipper. 2000. A representation for complex and evolving data dependencies in generation. In *Proc. of the 6th Conference on Applied Natural Language Processing*, pages 119–126, Seattle, Washington, USA, April.

Marie Meteer. 1991. Bridging the generation gap between text planning and linguistic realization. In *Computational Intelligence*, volume 7 (4).

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proc. of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 194–201, Stroudsburg, PA, USA.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87, March.

Ehud Reiter. 1994. Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? pages 163–170.

Eric K. Ringger, Michael Gamon, Robert C. Moore, David Rojas, Martine Smets, and Simon Corston-Oliver. 2004. Linguistically Informed Statistical Models of Constituent Structure for Ordering in Sentence Realization. In *Proc. of the 2004 International Conference on Computational Linguistics*, Geneva, Switzerland.

Jacques Robin. 1993. A revision-based generation architecture for reporting facts in their historical context. In *New Concepts in Natural Language Generation: Planning, Realization and Systems. Frances Pinter, London and*, pages 238–265. Pinter Publishers.

Michael Roth and Anette Frank. 2012. Aligning predicate argument structures in monolingual comparable texts: A new corpus for a new task. In *Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*SEM)*, Montreal, Canada.

Robert Rubinoff. 1992. Integrating text planning and linguistic choice by annotating linguistic structures. In Robert Dale, Eduard H. Hovy, Dietmar Rösner, and Oliviero Stock, editors, *NLG*, volume 587 of *Lecture Notes in Computer Science*, pages 45–56. Springer.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2010. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proc. of the 5th International Workshop on Semantic Evaluation*, pages 45–50, Uppsala, Sweden, July.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proc. of the 8th conference on International Language Resources and Evaluation*, Istanbul, Turkey, May.

Advaith Siddharthan and Ann Copestake. 2004. Generating referring expressions in open domains. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 407–414, Barcelona, Spain, July.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for nlp-assisted text annotation. In *Proc. of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April.

Leo Wanner, Simon Mille, and Bernd Bohnet. 2012. Towards a surface realization-oriented corpus annotation. In *Proc. of the 7th International Natural Language Generation Conference*, pages 22–30, Utica, IL, May.

Leo Wanner. 1994. Building another bridge over the generation gap. In *Proc. of the 7th International Workshop on Natural Language Generation*, INLG '94, pages 137–144, Stroudsburg, PA, USA.

Sina Zarrieß, Aoife Cahill, and Jonas Kuhn. 2011. Underspecifying and predicting voice for surface realisation ranking. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1007–1017, Portland, Oregon, USA, June.

Sina Zarrieß, Aoife Cahill, and Jonas Kuhn. 2012. To what extent does sentence-internal realisation reflect discourse context? a study on word order. In *Proc. of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 767–776, Avignon, France, April.