Inducing Document Plans for Concept-to-text Generation

Ioannis Konstas and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
ikonstas@inf.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In a language generation system, a content planner selects which elements must be included in the output text and the ordering between them. Recent empirical approaches perform content selection without any ordering and have thus no means to ensure that the output is coherent. In this paper we focus on the problem of generating text from a database and present a trainable end-to-end generation system that includes both content selection and ordering. Content plans are represented intuitively by a set of grammar rules that operate on the document level and are acquired automatically from training data. We develop two approaches: the first one is inspired from Rhetorical Structure Theory and represents the document as a tree of discourse relations between database records; the second one requires little linguistic sophistication and uses tree structures to represent global patterns of database record sequences within a document. Experimental evaluation on two domains yields considerable improvements over the state of the art for both approaches.

1 Introduction

Concept-to-text generation broadly refers to the task of automatically producing textual output from non-linguistic input (Reiter and Dale, 2000). Depending on the application and the domain at hand, the input may assume various representations including databases, expert system knowledge bases, simulations of physical systems, or formal meaning representations. Generation systems typically follow a pipeline architecture consisting of three components: *content planning* (selecting and ordering the parts of the input to be mentioned in the output text), *sentence planning* (determining the structure and lexical content of individual sentences), and *surface realization* (verbalizing the chosen content in natural language). Traditionally, these components are hand-engineered in order to ensure output of high quality.

More recently there has been growing interest in the application of learning methods because of their promise to make generation more robust and adaptable. Examples include learning which content should be present in a document (Duboue and McKeown, 2002; Barzilay and Lapata, 2005), how it should be aligned to utterances (Liang et al., 2009), and how to select a sentence plan among many alternatives (Stent et al., 2004). Beyond isolated components, a few approaches have emerged that tackle concept-to-text generation end-to-end. Due to the complexity of the task, most models simplify the generation process, e.g., by treating sentence planning and surface realization as one component (Angeli et al., 2010), by implementing content selection without any document planning (Konstas and Lapata, 2012; Angeli et al., 2010; Kim and Mooney, 2010), or by eliminating content planning entirely (Belz, 2008; Wong and Mooney, 2007).

In this paper we present a trainable end-to-end generation system that captures all components of the traditional pipeline, including document planning. Rather than breaking up the generation process into a sequence of local decisions, each learned separately (Reiter et al., 2005; Belz, 2008; Chen and Mooney, 2008; Kim and Mooney, 2010), our model performs content planning (i.e., document planning and content selection), sentence planning (i.e., lex-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1503–1514,
Seattle, Washington, USA, 18-21 October 2013.

Database Records
temp(time:6-21, min:**9**, mean:**15**, max:**21**) wind-spd(time:6-21, min:**15**, mean:**20**, max:**30**) sky-cover(time:6-9, percent:*25-50*) sky-cover(time:9-12, percent:*50-75*) wind-dir(time:6-21, mode:SSE) gust(time:6-21, min:**20**, mean:**30**, max:**40**)
Output Text
Cloudy, with a high around 20. South southeast wind between 15 and 30 mph. Gusts as high as 40 mph.

(a) WEATHERGOV

Database Records
desktop(cmd:lclick, name:*start*, type:button) start(cmd:lclick, name:*settings*, type:button) start-target(cmd:lclick, name:*control panel*, type:button) win-target(cmd:dblclick, name:*users and passwords*, type:item) contMenu(cmd:lclick, name:*advanced*, type:tab) action-contMenu(cmd:lclick, name:*advanced*, type:button)
Output Text
Click start, point to settings, and then click control panel. Double-click users and passwords. On the advanced tab, click advanced.

(b) WINHELP

Figure 1: Database records and corresponding text for (a) weather forecasting and (b) Windows troubleshooting. Each record has a type (e.g., win-target), and a set of fields. Each field has a value, which can be categorical (in typewriter), an integer (in **bold**), or a literal string (in *italics*).

icalization of input entries), and surface realization jointly. We focus on the problem of generating text from a database. The input to our model is a set of database records and collocated descriptions, examples of which are shown in Figure 1.

Given this input, we define a probabilistic context-free grammar (PCFG) that captures the structure of the database and how it can be verbalized. Specifically, we extend the model of Konstas and Lapata (2012) which also uses a PCFG to perform content selection and surface realization, but does not capture any aspect of document planning. We represent content plans with grammar rules which operate on the document level and are embedded on top of the original PCFG. We essentially learn a discourse grammar following two approaches. The first one is linguistically naive but applicable to multiple languages and domains; it extracts rules representing global patterns of record sequences within a sentence and among sentences from a training corpus. The second approach learns document plans based on *Rhetorical Structure Theory* (RST; Mann and Thomson, 1988); it therefore has a solid linguistic foundation, but is resource intensive as it assumes access to a text-level discourse parser.

We learn document plans automatically using both representations and develop a tractable decoding algorithm for finding the best output, i.e., derivation in our grammar. To the best of our knowledge, this is the first data-driven model to incorporate document planning in a joint end-to-end system. Experimental evaluation on the WEATHERGOV (Liang et al., 2009) and WINHELP (Branavan et al., 2009) domains shows that our approach improves over Konstas and Lapata (2012) by a wide margin.

2 Related Work

Content planning is a fundamental component in a natural generation system. Not only does it determine which information-bearing units to talk about, but also arranges them into a structure that creates coherent output. It is therefore not surprising that many content planners have been based on theories of discourse coherence (Hovy, 1993; Scott and de Souza, 1990). Other work has relied on generic planners (Dale, 1988) or schemas (Duboue and McKeown, 2002). In all cases, content plans are created manually, sometimes through corpus analysis. A few researchers recognize that this top-down approach to planning is too inflexible and adopt a generate-and-rank architecture instead (Mellish et al., 1998; Karamanis, 2003; Kibble and Power, 2004). The idea is to produce a large set of candidate plans and select the best one according to a ranking function. The latter is typically developed manually taking into account constraints relating to discourse coherence and the semantics of the domain.

Duboue and McKeown (2001) present perhaps the first empirical approach to content planning. They use techniques from computational biology to learn the basic patterns contained within a plan and the ordering among them. Duboue and McKeown (2002) learn a tree-like planner from an aligned corpus of semantic inputs and corresponding human-authored outputs using evolutionary al-

gorithms. More recent data-driven work focuses on end-to-end systems rather than individual components, however without taking document planning into account. For example, Kim and Mooney (2010) first define a generative model similar to Liang et al. (2009) that selects which database records to talk about and then use an existing surface realizer (Wong and Mooney, 2007) to render the chosen records in natural language. Their content planner has no notion of coherence. Angeli et al. (2010) adopt a more unified approach that builds on top of the alignment model of Liang et al. (2009). They break record selection into a series of locally coherent decisions, by first deciding on what records to talk about. Each choice is based on a history of previous decisions, which is encoded in the form of discriminative features in a log-linear model. Analogously, they choose fields for each record, and finally verbalize the input using automatically extracted domain-specific templates from training data.

Konstas and Lapata (2012) propose a joint model, which recasts content selection and surface realization into a parsing problem. Their model optimizes the choice of records, fields and words simultaneously, however they still select and order records locally. We replace their content selection mechanism (which is based on a simple markovized chaining of records) with global document representations. A *plan* in our model is identified either as a sequence of sentences, each containing a sequence of records, or as a tree where the internal nodes denote discourse information and the leaf nodes correspond to records.

3 Problem Formulation

The generator takes as input a set of database records $\mathbf{d}$ and outputs a text g that verbalizes some of these records. Each record token $r_i \in \mathbf{d}$, with $1 \leq i \leq |\mathbf{d}|$, has a type $r_i.t$ and a set of fields f associated with it. Fields have different values $f.v$ and types $f.t$ (i.e., integer, categorical, or literal strings). For example, in Figure 1b, win-target is a record type with three fields: cmd (denotes the action the user must perform on an object on their screen, e.g., left-click), name (denotes the name of the object), and type (denotes the type of the object). The values of these fields are `dblclick`, *users and passwords*, and `item`; name is a literal string, the rest are categorical.

During training, our algorithm is given a corpus consisting of several *scenarios*, i.e., database records paired with texts w (see Figure 1). For each scenario, the model first decides on a *global document plan*, i.e., it selects which *types of records* belong to each sentence (or phrase) and how these sentences (or phrases) should be ordered. Then it selects appropriate record tokens for each type and progressively chooses the most relevant fields; then, based on the values of the fields, it generates the final text, word by word.

Grammar Rules
1. $\mathrm{S} \rightarrow \mathrm{R}(\mathrm{start})$
2. $\mathrm{R}(r_i.t) \rightarrow \mathrm{FS}(r_j, start)\ \mathrm{R}(r_j.t) \mid \mathrm{FS}(r_j, start)$
3. $\mathrm{FS}(r, r.f_i) \rightarrow \mathrm{F}(r, r.f_j)\ \mathrm{FS}(r, r.f_j) \mid \mathrm{F}(r, r.f_j)$
4. $\mathrm{F}(r, r.f) \rightarrow \mathrm{W}(r, r.f)\ \mathrm{F}(r, r.f) \mid \mathrm{W}(r, r.f)$
5. $\mathrm{W}(r, r.f) \rightarrow \alpha \mid \mathrm{g}(f.v)$

Figure 2: Grammar G of the original model. Parentheses denote features, and impose constraints on the grammar.

4 Original Model

Our work builds on the model developed by Konstas and Lapata (2012). The latter is essentially a PCFG which captures both the structure of the input database and the way it renders into natural language. This grammar-based approach lends itself well to the incorporation of document planning which has traditionally assumed tree-like representations. We first briefly describe the original model and then present our extensions in Section 5.

Grammar Grammar G in Figure 2 defines a set of non-recursive CFG rewrite rules that capture the structure of the database, i.e., the relationship between records, records and fields, fields and words. These rules are domain-independent and could be applied to any database provided it follows the same structure. Non-terminal symbols are in capitals, the terminal symbol α corresponds to the vocabulary of the training set and $g(f.v)$ is a function which generates integers given the field value $f.v$. Note that all non-terminals have features (in parentheses) which

act as constraints and impose non-recursion (e.g., in rule (2) $i \neq j$, so that a record cannot emit itself).

Rule (1) defines the expansion from the start symbol S to the first record R of type *start*. The rules in (2) implement content selection, by choosing appropriate records from the database and generating a sequence. $\mathrm{R}(r_i.t)$ is the source record, $\mathrm{R}(r_j.t)$ is the target record and $\mathrm{FS}(r_j.start)$ is a place-holder symbol for the set of fields of record token r_j. This method is locally optimal, since it only keeps track of the previous type of record for each re-write. The rules in (3) conclude content selection on the field level, i.e., after we have chosen a record, we select and order the corresponding fields. Finally, the rules in (4) and (5) correspond to surface realization. The former rule binarizes the sequence of words emitted by a particular field $r.f$ in an attempt to capture local dependencies between words, such as multi-word expressions (e.g., right click, radio button). The latter rule defines the emission of words and integer numbers[1], given a field type and its value. Note that the original model lexicalizes field values of categorical and integer type only.

Training The rules of grammar G are associated with weights that are learned using the EM algorithm (Dempster et al., 1977). During training, the records, fields and values of database $\mathbf{d}$ and the words w from the associated text are observed, and the model learns the mapping between them. Notice that we use w to denote the gold-standard text and g to refer to the words generated by the model. The mapping between the database and the observed text is unknown and thus the weights of the rules define a hidden correspondence h between records, fields and their values.

Decoding Given a trained grammar G and an input scenario from a database $\mathbf{d}$, the model generates text by finding the most likely derivation, i.e., sequence of rewrite rules for the input. Although resembling parsing, the generation task is subtly different. In parsing, we observe a string of words and our goal is to find the most probable syntactic structure, i.e., hidden correspondence $\hat{h}$. In generation, however, the string is not observed; instead, we must find the best text $\hat{g}$, by maximizing both over h and g, where $g = g_1 \dots g_N$ is a sequence of words licensed by G. More formally:

$$\hat{g} = f\Big(\arg\max_{g,h} P\big((g,h)\big)\Big) \tag{1}$$

where f is a function that takes as input a derivation tree (g,h) and returns $\hat{g}$. Konstas and Lapata (2012) use a modified version of the CYK parser (Kasami, 1965; Younger, 1967) to find $\hat{g}$. Specifically, they intersect grammar G with a n-gram language model and calculate the most probable generation $\hat{g}$ as:

$$\hat{g} = f\Big(\arg\max_{g,h} p(g) \cdot p(g,h \,|\, \mathbf{d})\Big) \tag{2}$$

where $p(g,h \,|\, \mathbf{d})$ is the decoding likelihood for a sequence of words $g = g_1 \dots g_N$ of length N and the hidden correspondence h that emits it, i.e., the likelihood of the grammar for a given database input scenario $\mathbf{d}$. $p(g)$ is a measure of the quality of each output and is provided by the n-gram language model.

5 Extensions

In this section we extend the model of Konstas and Lapata (2012) by developing two more sophisticated content selection approaches which are informed by a global plan of the document to be generated.

5.1 Planning with Record Sequences

Grammar Our key idea is to replace the content selection mechanism of the original model with a *document plan* which essentially defines a grammar on record types. We split a document into sentences, each terminated by a full-stop. Then a sentence is further split into a sequence of record types. Contrary to the original model, we observe a complete sequence[2] of record types, split into sentences. This way we learn domain-specific patterns of frequently occurring record type sequences among the sentences of a document, as well as more local structures within a sentence. We thus substitute rules (1)–(2) in Figure 2 with sub-grammar G_{RSE} based on record type sequences:

Definition 1 (G_{RSE} grammar)

$$G_{RSE} = \{\Sigma_R,\, N_{RSE},\, P_{RSE},\, D\}$$

[1] The function $g(f.v) : \mathbb{Z} \rightarrow \mathbb{Z}$, generates an integer in the following six ways (Liang et al., 2009): identical, rounding up/down to a multiple of 5, rounding off a multiple of 5 and adding or subtracting some noise modelled by a geometric distribution.

[2] Note that a sequence is different from a permutation, as we may allow repetitions or omissions of certain record types.

where Σ_R is a set of terminal symbols $R(r.t)$, and N_{RSE} is a set of non-terminal symbols:

$$N_{RSE} = \{D, SENT\}$$

where D represents the start symbol and $SENT$ a sequence of records. P_{RSE} is a set of production rules of the form:

(a) $D \rightarrow SENT(t_i, \ldots, t_j) \ \ldots \ SENT(t_l, \ldots, t_m)$

(b) $SENT(t_i, \ldots, t_j) \rightarrow R(r_a.t_i) \ \ldots \ R(r_k.t_j) \cdot$

where t is a record type, t_i, t_j, t_l and t_m may overlap and r_a, r_k are record tokens of type t_i and t_j respectively. The corresponding weights for the production rules P_{RSE} are:

Definition 2 (G_{RSE} weights)

(a) $p(t_i, \ldots, t_j, \ldots t_l, \ldots, t_m \mid D)$

(b) $p(t_i) \cdot \ldots \cdot p(t_j) = \frac{1}{|\mathbf{s}(t_i)|} \cdot \ldots \cdot \frac{1}{|\mathbf{s}(t_j)|}$

where $\mathbf{s}(t)$ is a function that returns the set of records with type t (Liang et al., 2009).

Rule (a) defines the expansion from the start symbol D to a sequence of sentences, each represented by the non-terminal $SENT$. Similarly to the original grammar G, we employ the use of features (in parentheses) to denote a sequence of record types. The same record types may recur in different sentences, but not in the same one. The weight of rule (a) is simply the joint probability of all the record types present, ordered and segmented appropriately into sentences in the document, given the start symbol.

Once record types have been selected (on a per sentence basis) we move on to rule (b) which describes how each non-terminal $SENT$ expands to an ordered sequence of records R, as they are observed within a sentence (see the terminal symbol '.' at the end of the rule). Notice that a record type t_i may correspond to several record tokens r_a. Rules (3)–(5) in grammar G make decisions on these tokens based on the overall content of the database and the field/value selection. The weight of this rule is the product of the weights of each record type. This is set to the uniform distribution over $\{1, ..., |\mathbf{s}(t)|\}$ for record type t, where $|\mathbf{s}(t)|$ is the number of records with that type.

Figure 3d shows an example tree for the database input in Figure 1b, using G_{RSE} and assuming that the alignments between records and text are given. The top level of the tree refers to the sequence of record types as they are observed in the text. The first sentence contains three records with types 'desktop', 'start' and 'start-target', each corresponding to the textual segments *click start*, *point to settings*, *and then click control panel*. The next level on the tree, denotes the choice of record tokens for each sentence, provided that we have decided on the choice and order of their types (see Figure 3b). In Figure 3d, the bottom-left sub-tree corresponds to the choice of the first three records of Figure 1b.

Training A straightforward way to train the extended model would be to embed the parameters of G_{RSE} in the original model and then run the EM algorithm using inside-outside at the E-step. Unfortunately, this method will induce a prohibitively large search space. Rule (a) enumerates all possible combinations of record type sequences and the number grows exponentially even for a few record types and a small sequence size. To tackle this problem, we extracted rules for G_{RSE} from the training data, based on the assumption that there will be far fewer unique sequences of record types per dataset than exhaustively enumerating all possibilities.

For each scenario, we obtain a word-by-word alignment between the database records and the corresponding text. In our experiments we used Liang et al.'s (2009) unsupervised model, however any other semi- or fully supervised method could be used. As we show in Section 7, the quality of the alignment inevitably correlates with the quality of the extracted grammar and the decoder's output. We then map the aligned record tokens to their corresponding types, merge adjacent words with the same type and segment on punctuation (see Figure 3b). Next, we create the corresponding tree according to G_{RSE} (Figure 3d) and binarize it. We experimented both with left and right binarization and adhered to the latter, as it obtained a more compact set of rules. Finally, we collectively count the rule weights on the resulting treebank and extract a rule set, discarding rules with frequency less than three.

Using the extracted (weighted) G_{RSE} rules, we run the EM algorithm via inside-outside and learn the weights for the remaining rules in G. Decoding remains the same as in Konstas and Lapata (2012); the only requirement is that the extracted grammar remains binarized in order to guarantee the cubic

desktop$_1$	start$_1$	start-target$_1$	win-target$_1$	contMenu$_1$	action-contMenu$_1$
Click start,	point to settings,	and then click control panel.	Double-click users and passwords.	On the advanced tab ,	click advanced.

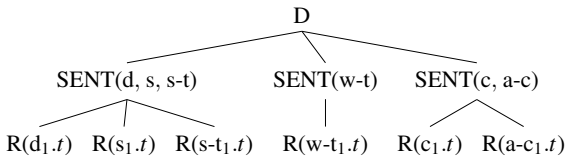
(a) Record token alignments

[desktop start start-target‖win-target‖contMenu action-contMenu‖]

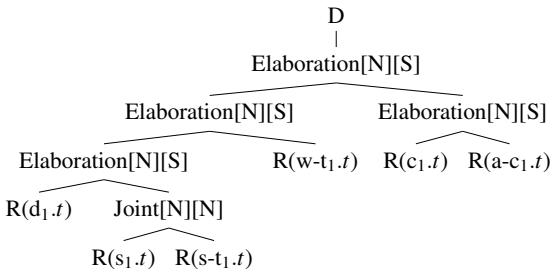
(b) Record type segmentation

[Click start,]$^{desktop_1.t}$ [point to settings,]$^{start_1.t}$ [and then click control panel.]$^{start-target_1.t}$ [Double-click users and passwords.]$^{win-target_1.t}$ [On the advanced tab,]$^{contMenu_1.t}$ [click advanced.]$^{action-contMenu_1.t}$

(c) Segmentation of text into EDUs



(d) Document plan using the G_{RSE} grammar



(e) Document plan using the G_{RST} grammar

Figure 3: Grammar extraction example from the WINHELP domain using G_{RSE} and G_{RST}. For G_{RSE}, we take the alignments of records on words and map them to their corresponding types (a); we then segment record types into sentences (b); and finally, create a tree using grammar G_{RSE} (c). For G_{RST}, we segment the text into EDUs based on the records they align to (d) and output the discourse tree (omitted here for brevity's sake); we build the document plan once we substitute the EDUs with their corresponding record types (e).

bound of the Viterbi search algorithm. Note that the original grammar is limited to the generation of categorical and integer values. We extend it to support the generation of strings. The following rule adds a simple verbatim lexicalization for string values:

$$\mathrm{W}(r, r.f) \rightarrow gen\_str(f.v, i)$$
$$gen\_str(f.v, i) : V \rightarrow V,\ f.v \in V$$

where V is the set of words for the fields of type string, and gen_str is a function that takes the value of a string-typed field $f.v$, and the position i in the string, and generates the corresponding word at that position. For example, gen_str(*users and passwords*, 3) = *passwords*. The weight of this rule is set to 1.

5.2 Planning with Rhetorical Structure Theory

Grammar RST (Mann and Thompson, 1988) is a theory of text organization which provides a framework for analyzing text. A basic tenet of the theory is that a text consists of hierarchically organized text spans or elementary discourse units (EDUs) that stand in various relations to one another (e.g., *Elaboration*, *Attribution*). These "rhetorical relations" hold between two adjacent parts of the text, where typically, one part is "nuclear" and one a "satellite". An analysis of a text consists in identifying the relations holding between successively larger parts of the text, yielding a natural hierarchical description of the rhetorical organization of the text. From its very inception, RST was conceived as a way to characterize text and textual relations for the purpose of text generation.

In order to create a RST-inspired document plan for our input (i.e., database records paired with texts), we make the following assumption: each record corresponds to a unique non-overlapping span in the collocated text, and can be therefore mapped to an EDU. Assuming the text has been segmented and aligned to a sequence of records, we can create a discourse tree with record types (in place of their corresponding EDUs) as leaf nodes. Again, we define a sub-grammar G_{RST} which replaces rules (1)–(2) from Figure 2:

Definition 3 (G_{RST} grammar)

$$G_{RST} = \{\Sigma_R, N_{RST}, P_{RST}, D\}$$

where Σ_R is the alphabet of leaf nodes as defined in Section 5.1, N_{RST} is a set of non-terminals corresponding to rhetorical relations augmented with nucleus-satellite information (e.g., *Elaboration[N][S]* stands for the elaboration relation between the nucleus EDU left-adjoining with the satellite EDU), P_{RST} is the set of production rules of the form $P_{RST} \subseteq N_{RST} \times \{N_{RST} \cup \Sigma_R\} \times \{N_{RST} \cup \Sigma_R\}$ associated with a weight for each rule, and $D \in N_{RST}$ is the root symbol. Figure 3e gives the discourse tree for the database input of Figure 1b, using G_{RST}.

Training In order to obtain the weighted productions of G_{RST}, we use an existing state-of-the-art discourse parser[3] (Feng and Hirst, 2012) trained on the RST-DT corpus (Carlson et al., 2001). The latter contains a selection of 385 Wall Street Journal articles which have been annotated using the framework of RST and an inventory of 78 rhetorical relations, classified into 18 coarse-grained categories (Carlson and Marcu, 2001). Figure 4 gives a comparison of the distribution of relations extracted for the two datasets we used, against the gold-standard annotation of RST-DT. The statistics for the RST-DT corpus are taken from Williams and Power (2008). The relative frequencies of relations on both datasets follow closely the distribution of those in RST-DT, thus empirically supporting the application of the RST framework to our data.

We segment each document in our training set into EDUs based on the record-to-text alignments given by the model of Liang et al. (2009) (see Figure 3c). We then run the discourse parser on the resulting EDUs, and retrieve the corresponding discourse tree; the internal nodes are labelled with one of the RST relations. Finally, we replace the leaf EDUs with their respective terminal symbols $R(r.t) \in \Sigma_R$ (Figure 3e) and collect the resulting grammar productions; their weights are calculated via maximum likelihood estimation based on their collective counts in the parse trees licensed by G_{RST}. Training and decoding of the extended generation model (after we embed G_{RST} in the original grammar G) is performed identically to Section 5.1.

[3]Publicly available from `http://www.cs.toronto.edu/~weifeng/software.html`.

6 Experimental Design

Data Since our aim was to evaluate the planning component of our model, we used datasets whose documents are at least a few sentences long. Specifically, we generated weather forecasts and troubleshooting guides for an operating system. For the first domain (henceforth WEATHERGOV) we used the dataset of Liang et al. (2009), which consists of 29,528 weather scenarios for 3,753 major US cities (collected over four days). The database has 12 record types, each scenario contains on average 36 records, 5.8 out of which are mentioned in the text. A document has 29.3 words and is four sentences long. The vocabulary is 345 words. We used 25,000 scenarios from WEATHERGOV for training, 1,000 scenarios for development and 3,528 scenarios for testing.

For the second domain (henceforth WINHELP) we used the dataset of Branavan et al. (2009), which consists of 128 scenarios. These are articles from Microsoft's Help and Support website[4] and contain step-by-step instructions on how to perform tasks on the Windows 2000 operating system. In its original format, the database provides a semantic representation of the textual guide, i.e., it represents the user's actions on the operating system's UI. We semi-automatically converted this representation into a schema of records, fields and values, following the conventions adopted in Branavan et al. (2009).[5] The final database has 13 record types. Each scenario has 9.2 records and each document 51.92 words with 4.3 sentences. The vocabulary is 629 words. We performed 10-fold cross-validation on the entire dataset for training and testing. Compared to WEATHERGOV, WINHELP documents are longer with a larger vocabulary. More importantly, due to the nature of the domain, i.e., giving instructions, content selection is critical not only in terms of what to say but also in what order.

Grammar Extraction and Parameter Setting We obtained alignments between database records and textual segments for both domains and grammars (G_{RSE} and G_{RST}) using the unsupervised model of Liang et al. (2009). On WEATHERGOV, we extracted a G_{RSE} grammar with 663 rules (after bi-

[4]`support.microsoft.com`

[5]The dataset can be downloaded from `http://homepages.inf.ed.ac.uk/ikonstas/index.php?page=resources`

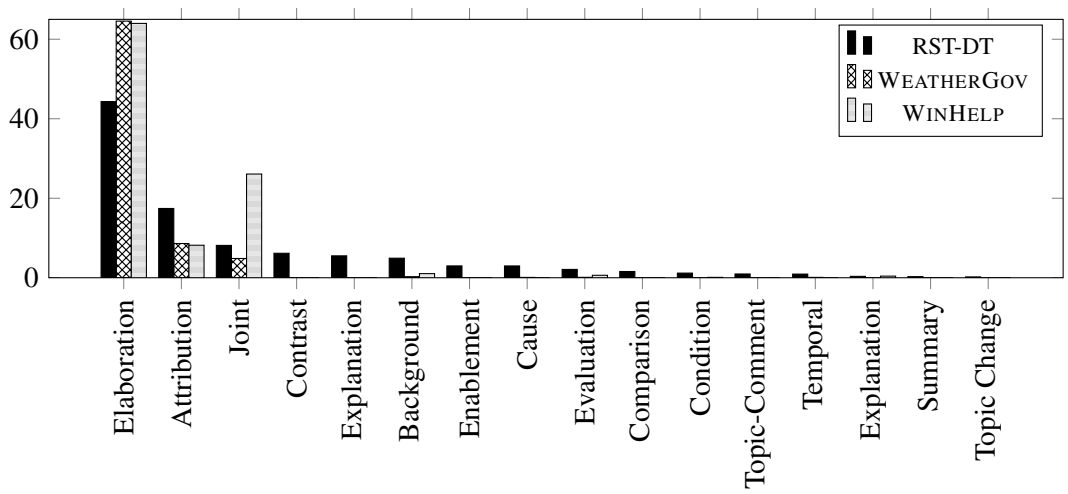


Figure 4: Distribution of RST relations on WEATHERGOV, WINHELP, and the RST-DT (Williams and Power, 2008).

narization). The WINHELP dataset is considerably smaller, and as a result the procedure described in Section 5.1 yields a very sparse grammar. To alleviate this, we *horizontally markovized* the right-hand side of each rule (Collins, 1999; Klein and Manning, 2003).[6] After markovization, we obtained a G_{RSE} grammar with 516 rules. On WEATHERGOV, we extracted 434 rules for G_{RST}. On WINHELP we could not follow the horizontal markovization procedure, since the discourse trees are already binarized. Instead, we performed *vertical markovization*, i.e., annotated each non-terminal with their parent node (Johnson, 1998) and obtained a G_{RST} grammar with 419 rules. The model of Konstas and Lapata (2012) has two parameters, namely the number of k-best lists to keep in each derivation, and the order of the language model. We tuned k experimentally on the development set and obtained best results with 60 for WEATHERGOV and 120 for WINHELP. We used a trigram model for both domains, trained on each training set.

Evaluation We compared two configurations of our system, one with a content planning component based on record type sequences (G_{RSE}) and another one based on RST (G_{RST}). In both cases content plans were extracted from (noisy) unsupervised alignments. As a baseline, we used the original model of Konstas and Lapata (2012). We also compared our model to Angeli et al.'s system (2010), which is state of the art on WEATHERGOV.

System output was evaluated automatically, using the BLEU modified precision score (Papineni et al., 2002) with the human-written text as reference. In addition, we evaluated the generated text by eliciting human judgments. Participants were presented with a scenario and its corresponding verbalization and were asked to rate the latter along three dimensions: fluency (is the text grammatical?), semantic correctness (does the meaning conveyed by the text correspond to the database input?) and coherence (is the text comprehensible and logically structured?). Participants used a five point rating scale where a high number indicates better performance. We randomly selected 12 documents from the test set (for each domain) and produced output with the system of Konstas and Lapata (2012) (henceforth K&L), our two models using G_{RSE} and G_{RST}, respectively, and Angeli et al. (2010) (henceforth ANGELI). We also included the original text (HUMAN) as gold-standard. We obtained ratings for 60 (12×5) scenario-text pairs for each domain. Examples of the documents shown to the participants are given in Table 1.

The study was conducted over the Internet us-

[6] When horizontally markovizing, we can encode an arbitrary amount of context in the intermediate non-terminals that result from this process; in our case we store h=1 horizontal siblings plus the mother left-hand side (LHS) non-terminal, in order to uniquely identify the Markov chain. For example, $A \rightarrow B\ C\ D$ becomes $A \rightarrow B\ \langle A \ldots B \rangle$, $\langle A \ldots B \rangle \rightarrow C\ \langle A \ldots C \rangle$, $\langle A \ldots C \rangle \rightarrow D$.

	WEATHERGOV	WINHELP
G_{RSE}	Showers before noon. Cloudy, with a high near 38. Southwest wind between 3 and 8 mph. Chance of precipitation is 55 %.	Right-click my network places, and then click properties. Right-click local area connection, and click properties. Click to select the file and printer sharing for Microsoft networks, and then click ok.
G_{RST}	Showers likely. Mostly cloudy, with a high around 38. South wind between 1 and 8 mph. Chance of precipitation is 55 %.	Right-click my network places, and then click properties. Right-click local area connection. Click file and printer sharing for Microsoft networks, and click ok.
K&L	A chance of showers. Otherwise, cloudy, with a high near 38. Southwest wind between 3 and 8 mph.	Right-click my network places, click properties. Right-click local area connection. Click to select the file and printer sharing for Microsoft networks, and then click ok.
ANGELI	A chance of rain or drizzle after 9am. Mostly cloudy, with a high near 38. Southwest wind between 3 and 8 mph. Chance of precipitation is 50 %.	Right-click my network places, and then click properties on the tools menu, and then click properties. Right-click local area connection, and then click properties. Click file and printer sharing for Microsoft networks, and then click ok.
HUMAN	A 50 percent chance of showers. Cloudy, with a high near 38. Southwest wind between 3 and 6 mph.	Right-click my network places, and then click properties. Right-click local area connection, and then click properties. Click to select the file and printer sharing for Microsoft networks check box. Click ok.

Table 1: Human-authored text and system output on WEATHERGOV and WINHELP.

ing Amazon Mechanical Turk[7], and involved 200 volunteers (100 for WEATHERGOV, and 100 for WINHELP), all self reported native English speakers. For WINHELP, we made sure participants were computer-literate and familiar with the Windows operating system by administering a short questionnaire prior to the experiment.

7 Results

The results of the automatic evaluation are summarized in Table 2. Overall, our models outperform K&L's system by a wide margin on both datasets. The two content planners (G_{RSE} and G_{RST}) perform comparably in terms of BLEU. This suggests that document plans induced solely from data are of similar quality to those informed by RST. This is an encouraging result given that RST-style discourse parsers are currently available only for English. ANGELI performs better on WEATHERGOV possibly due to better output quality on the surface level. Their system defines trigger patterns that specifically lexicalize record fields containing numbers. In contrast, on WINHELP it is difficult to explicitly specify such patterns, as none of the record fields are numeric; as a result their system performs poorly compared to the other models.

To assess the impact of the alignment on the content planner, we also extracted G_{RSE} from cleaner alignments which we obtained automatically via human-crafted heuristics for each domain. The heuristics performed mostly anchor matching between database records and words in the text (e.g., the value `Lkly` of the field rainChance, matches with the string *rain likely* in the text). Using these alignments, G_{RSE} obtained a BLEU score of 39.23 on WEATHERGOV and 41.35 on WINHELP. These results indicate that improved alignments would lead to more accurate grammar rules. WEATHERGOV seems more sensitive to the alignments than WINHELP. This is probably because the dataset shows more structural variations in the choice of record types at the document level, and therefore the grammar extracted from the unsupervised alignments is noisier. Unfortunately, performing this kind of analysis for G_{RST} would require gold standard segmentation of our training corpus into EDUs which we neither have nor can easily approximate via heuristics.

The results of our human evaluation study are shown in Table 3. We carried out an Analysis of Variance (ANOVA) to examine the effect of system

[7] https://www.mturk.com

Model	WEATHERGOV	WINHELP
G_{RSE}	35.60	40.92
G_{RST}	36.54	40.65
K&L	33.70	38.26
ANGELI	38.40	32.21

Table 2: Automatic evaluation of system output using BLEU-4.

	WEATHERGOV			WINHELP		
Model	FL	SC	CO	FL	SC	CO
G_{RSE}	4.25	3.75	4.18	3.59	3.21	3.35
G_{RST}	4.10	3.68	4.10	3.45	3.29	3.22
K&L	3.73	3.25	3.59	3.27	2.97	2.93
ANGELI	3.90	3.44	3.82	3.44	2.79	2.97
HUMAN	4.22	3.72	4.11	4.20	4.41	4.25

Table 3: Mean ratings for fluency (FL), semantic correctness (SC) and coherence (CO) on system output elicited by humans.

type (G_{RSE}, G_{RST}, K&L, ANGELI, and HUMAN) on fluency, semantic correctness and coherence ratings. Means differences of 0.2 or more are significant at the 0.05 level using a post-hoc Tukey test. Interestingly, we observe that document planning improves system output overall, not only in terms of coherence. Across all dimensions our models are perceived better than K&L and ANGELI. As far as coherence is concerned, the two content planners are rated comparably (differences in the means are not significant). Both G_{RSE} and G_{RST} are significantly better than the comparison systems (ANGELI and K&L). Table 1 illustrates examples of system output along with the gold standard content selection for reference, for the WEATHERGOV and WINHELP domains, respectively.

In sum, we observe that integrating document planning either via G_{RSE} or G_{RST} boosts performance. Document plans induced from record sequences exhibit similar performance, compared to those generated using expert-derived linguistic knowledge. Our systems are consistently better than K&L both in terms of automatic and human evaluation and are close or better than the supervised model of Angeli et al. (2010). We also show that feeding the system with a grammar of better quality can achieve state-of-the-art performance, without further changes to the model.

8 Conclusions

In this paper, we have proposed an end-to-end system that generates text from database input and captures all components of the traditional generation pipeline, including document planning. Document plans are induced automatically from training data and are represented intuitively by PCFG rules capturing the structure of the database and the way it renders to text. We proposed two complementary approaches to inducing content planners. In a first linguistically naive approach, a document is modelled as a sequence of sentences and each sentence as a sequence of records. Our second approach draws inspiration from Rhetorical Structure Theory (Mann and Thomson, 1988) and represents a document as a tree with intermediate nodes corresponding to discourse relations, and leaf nodes to database records.

Experiments with both approaches demonstrate improvements over models that do not incorporate document planning. In the future, we would like to tackle more challenging domains, such as NFL recaps, financial articles and biographies (Howald et al., 2013; Schilder et al., 2013). Our models could also benefit from the development of more sophisticated planners either via grammar refinement or more expressive grammar formalisms (Cohn et al., 2010).

Acknowledgments

We are grateful to Percy Liang and Gabor Angeli for providing us with their code and data. Thanks to Giorgio Satta and Charles Sutton for helpful comments and suggestions. We also thank the members of the Probabilistic Models reading group at the University of Edinburgh for useful feedback.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology and*

Empirical Methods in Natural Language Processing, pages 331–338, Vancouver, British Columbia.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, Suntec, Singapore.

L. Carlson and D. Marcu. 2001. Discourse tagging reference manual. Technical report, Univ. of Southern California / Information Sciences Institute.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of International Conference on Machine Learning*, pages 128–135, Helsinki, Finland.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 11(November):3053–3096.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Robert Dale. 1988. *Generating referring expressions in a domain of objects and processes*. Ph.D. thesis, University of Edinburgh.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society, series B*, 39(1):1–38.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 172–179.

Pablo A. Duboue and Kathleen R. McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of International Natural Language Generation*, pages 89–96, Ramapo Mountains, NY.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 60–68, Jeju Island, Korea.

Eduard Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385.

Blake Howald, Ravikumar Kondadadi, and Frank Schilder. 2013. Domain adaptable semantic clustering in statistical nlg. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 143–154, Potsdam, Germany, March. Association for Computational Linguistics.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, December.

Nikiforos Karamanis. 2003. *Entity Coherence for Descriptive Text Structuring*. Ph.D. thesis, University of Edinburgh.

Tadao Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA.

Rodger Kibble and Richard Power. 2004. Optimising referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

Joohyun Kim and Raymond Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd Conference on Computational Linguistics*, pages 543–551, Beijing, China.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430. Association for Computational Linguistics Morristown, NJ, USA.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761, Montréal, Canada.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

William C. Mann and Sandra A. Thomson. 1988. Rhetorical structure theory. *Text*, 8(3):243–281.

Chris Mellish, Alisdair Knott, Jon Oberlander, and Mick O'Donnell. 1998. Experiments using stochastic search for text planning. In *Proceedings of International Natural Language Generation*, pages 98–107, New Brunswick, NJ.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

Frank Schilder, Blake Howald, and Ravi Kondadadi. 2013. Gennext: A consolidated domain adaptable nlg system. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 178–182, Sofia, Bulgaria, August. Association for Computational Linguistics.

Donia Scott and Clarisse Sieckenius de Souza. 1990. Getting the message across in RST-based text generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, New York.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of Association for Computational Linguistics*, pages 79–86, Barcelona, Spain.

Sandra Williams and Richard Power. 2008. Deriving rhetorical complexity data from the rst-dt corpus. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, May.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of the Human Language Technology and the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–179, Rochester, NY.

Daniel H Younger. 1967. Recognition and parsing for context-free languages in time n^3. *Information and Control*, 10(2):189–208.