A Classification-driven Approach to Document Planning

Rafael L. de Oliveira, Eder M. de Novais, Roberto P. A. de Araujo, Ivandré Paraboni
University of São Paulo (USP)
School of Arts, Sciences and Humanities (EACH)
Av. Arlindo Bettio, 1000 - São Paulo, Brazil
{ rafaellage, eder.novais, roberto.araujo, ivandre } @usp.br

Abstract

Document Planning - the task of deciding which content messages should be realised in a target document based on raw data provided by an underlying application, and how these messages should be structured - is arguably one of the most crucial tasks in Natural Language Generation (NLG). In this work we present a machine learning approach to Document Planning that is entirely trainable from annotated corpora, and which paves the way to our long-term goal of developing a text generator system based on a series of classifiers for a simple NLG application in the education domain.

Keywords

Document Planning; Content Selection.

1. Introduction

Natural Language Generation (NLG) systems are used whenever simple, 'canned' text is not sufficient, and greater (i.e., closer to human performance) linguistic variation is required. The traditional NLG architecture is often depicted in simplified form as a 3-stages pipelined process (Document Planning, Sentence Planning and Surface Realisation, cf. [1]), a division that is at least partially motivated by the sheer complexity of the task. Starting from a high-level communicative goal of describing a given domain concept, the system builds up a plan to represent the input data up to the point in which fully-specified text in natural language is produced. The Document Planning module is responsible for deciding what information to communicate (this being the task of *Content Determination*) and then how this information should be structured for presentation (this being the task of *Document Structuring*.) For a more comprehensive discussion of the role of Document Planning and its subtasks in the NLG architecture we report to [3].

Document Planning is arguably one of the most crucial components of an NLG system [1]: if a generated document presents the required information in a reasonably coherent structure, then the system may be considered successful even if the text shows surface flaws or limited linguistic variation. On the other hand, if the required information is missing from the text, or if the text is poorly structured, then the overall results are most likely unsatisfactory regardless of how well the individual sentences were realised.

When speaking of Data-to-Text generation[1], Document Planning is often preceded by a *Data Interpretation* stage [2] that processes raw data application in the first place. In what follows, we discuss the early stages of a simple Data-to-Text NLG application addressing some aspects of both issues, namely, which chunks of information – or messages - should be included in the generated text from the raw data provided by an underlying application, and how such messages should be structured within a standard RST framework [4]. We will argue that at least in simple NLG applications, some of these issues may be tackled using trainable and (at least partially) domain-independent methods. The focus of this paper is one such method, in which we apply standard machine learning techniques to both Content Determination and Document Structuring, and which can be viewed as a first step towards the development of a trainable text-generating application based on a series of classifiers.

2. Application and Training Data

We envisage a simple NLG application in which grades obtained by University students in a given course are described as short reports generated automatically from raw data (i.e., the numeric grades themselves) available from their academic records. Thus, the input to our system will be a student's record, and the output is a report conveying a series of statements such as "You fared well in the regular exams and your grades on this subject were above the average of your class" etc. Such reports can be useful to both students keen to learn how their professors interpret their efforts, and to the professors themselves who may have an at-a-glance view of the student's progress.

A substantial part of our work consisted of preparing training data. We started by collecting 241 records of students' academic performance data in five courses taught by a single professor (who can be viewed as the domain expert) in an academic term. Each record consists of a set of 25 values representing various aspects of a student's academic performance: figures about attendance records, examination grades at various stages throughout the course, and the average grades obtained by the entire class in the same examinations. Additional attributes describe how the available grades should be interpreted in that particular course or term (e.g., whether a given practical exercise was compulsory etc.) From these data we intend to generate textual descriptions of both what each student achieved individually, and how their performance compares to their peers'.

[1] For a large-scale application of this kind, see [5].

International Conference RANLP 2009 - Borovets, Bulgaria, pages 324–329

For each one of the 241 data records, the domain expert has also authored a short (about 5-sentences long, and to some extent normalised) sample report conveying a series of statements about the overall progress of the student. The reports are entirely purpose-made, i.e., written so as to provide training data for a machine-learned NLG application. Similar methodology has been employed in an NLG system (also in the education domain) described in [6], and it contrasts the use of naturally-occurring texts as a model for the application.

Unlike common practice in many domain-independent NLP tasks, we have collected aligned data-text instances produced by a *single author*. In the present case this was necessary because we are interested in establishing the mappings from raw data (e.g., students' grades) to semantics (i.e., the interpretation of the data according to the professor in charge), and which may vary wildly across domain experts[2]. The fact that we are dealing with a domain- and author-dependent problem should not be viewed as unappealing to the wide research community, though: as our approach is intended to be trainable from a collection of text-data alignments, our work remains in principle adaptable to a particular author or domain, as we will discuss later.

As suggested in [7], the collected reports were manually segmented and annotated with information about the meanings or content messages that they intend to convey[3]. In doing so, we faced the question of how these messages should be defined: on the one hand, content messages could be sufficiently detailed so as to represent the meaning of atomic text units (e.g., single words.) On the other hand, meanings could span over entire sentences or even paragraphs. As pointed out in [1], the level of granularity of content messages should presumably be determined by the expected linguistic variation of the output text. In our case, given the regularity of our target documents, each text was simply segmented in meaningful units from which the corresponding messages were readily identified. The resulting list of messages and the segmentation scheme were then refined for completeness, and infrequent instances were eliminated (which of course reduced the possible linguistic variation of the output.) As a result, the possible contents of each document could be modelled as a 14-messages vector represented in flat semantics as attribute-value pairs, and each text segment in the document was annotated with one such message.

Put together, data and corresponding reports make a complete training data set for corpus-based NLG that we have called the *SINotas* corpus. The corpus consists of a structured collection of the above 241 data-text alignments annotated in XML format, including basic sentence segmentation (provided at the message level only, as discussed above), part-of-speech information and partial discourse structure represented as manually annotated RST relations [4].

The *SINotas* corpus is a valuable NLG resource in its own right, and a ready-to-use testbed for NLG research in Portuguese and related languages. However, as we have abstracted away from the application raw data by modelling the underlying semantics as content messages, *SINotas* does not convey the kind of low-level representation available from, e.g., the SUMTIME-METEO corpus described in [8], which aligns text directly with domain data[4].

3. Document Planning as Classification

We will use the *SINotas* data-text aligned corpus described in the previous section to develop a number of modules of a simple corpus-based NLG system as a series of classifiers, using off-the-shelf learning algorithms. Serialised classifiers have been applied to other NLG tasks, e.g., surface realisation as in [9,10][5].

Regarding related work in the field, we notice that the early stages of Document Planning (and particularly, Content Determination issues) seem to be somewhat misrepresented in the NLG literature, a gap that might be explained by the domain-dependent nature of the task (i.e., the dealing with raw application data.) Content Determination has been performed using statistical techniques in [11], followed by a machine learning approach to select relevant information. The same general principal is applied in [12] in the domain of American football matches, and taking contextual dependencies into account in a so-called 'collective' content selection approach. An extension of this work has been recently presented in [13] for the domain of cricket game with a novel alignment technique. In all these cases, the main focus is the automatic data-text alignment (which in our case was performed manually via corpus annotation) and they do not address our second subtask, Document Structuring.

3.1 Content Determination

Content Determination can be viewed as the task of computing content messages (e.g., in the form of predicate-argument structures) from the input data provided by the underlying application. In our work this is implemented as a 2-steps process: first, we compute all possible messages derivable from the application data (a task that can be viewed as a simplified form of *data interpretation* as in [2]) and then we select the subset of

[2] Had we mixed data produced by various authors in a single training set, it would not be possible to establish meaningful data-text mappings. For example, a grade 5.0 may be rated as 'good' by a particular professor, but simply as 'poor' by a less benign one.

[3] For an example of automatic alignment technique applicable to this task see [12].

[4] In other words, *SINotas* does not contain personal data (e.g., student's grades) but simply text and semantic features derived from them for research purposes.

[5] For instance, the system *Amalgam* described in [10] uses a series of 18 decision-trees to implement various tasks ranging from lexical choice to punctuation.

messages that should actually be realised in a particular document (which we will call *content selection.*) We will discuss each step in turn.

Data interpretation is performed as follows: given the 25 input values produced by the application, we intend to produce the set of 14 messages representing the semantic contents of the target document (some of which possibly conveying '*null*' values that stand for missing or irrelevant information.) This procedure was implemented by means of a classifier in which the 25 input values are learning features to each of the 14 output classes (annotated as messages in the *SINotas* corpus.)

For example, given a low grade in the examinations we would expect the corresponding *provas_aval* class to be assigned a negative value such as "*insufficient*". Using the application data and the *SINotas* corpus, 241 * 14 = 3374 training instances of data interpretation were used to classify each of the 14 output messages as below[6]:

$[val_1, val_2 ... val_{25}, message_1]$
...
$[val_1, val_2 ... val_{25}, message_{14}]$

The second Content Determination task – content selection - consists of deciding which of the generated messages should end up realised as surface text. This is necessary because not all available information appears in the output, that is, different value combinations may result in different reports altogether, and some values that are highly prominent in one context may be even discarded in other situations in which different information should be spelled out. For example, good overall results may make a single low grade not worth mentioning at all. Similarly, a student that has decided not to sit the final exams does not need to be told that his/her grades were 'below average' etc.

Given as an input the original 14-messages vector, we would like to filter out irrelevant content based on what is actually shown (or not shown) in the sample output texts as seen in the corpus. To this end, we defined a set of 14 learning features comprising the previously generated messages and 14 binary classes representing whether each of them were actually realised as text (true) or simply omitted (false). Once again, 241 training instances of content selection were extracted from the corpus to classify each of the 14 '*realise'* binary classes) making 3374 instances as below:

$[msg_1, msg_2 ... msg_{14}, realise\_msg_1]$
...
$[msg_1, msg_2 ... msg_{14}, realise\_msg_{14}]$

Each classification task was performed individually, that is, we did not use the reminder (13) binary classes as learning features to each class. This may in principle seem counter-intuitive, as the textual realisation of one message could hinge on whether others are realised or not, but such dependencies were not observed in our data. By contrasts, see for instance the collective selection approach in [12].

[6] In practice, however, none of the classification tasks required the use of all 25 learning features, as we discuss in section 5.

3.2 Document Structuring

We are interested in two particular aspects of Document Structuring (and which to some extent cover aspects of Microplanning in the standard pipeline NLG architecture in [1] as well): the task of organising the content messages computed in the previous Content Determination stage into sentences, and then organising these sentences in a global rhetorical structure. Both tasks consist of computing RST relations between content messages, in the first case within sentences (which we will call *within-sentence structuring*) and in the second case between sentences (called *between-sentences structuring*.) In our classification-driven approach this will be performed in a bottom-up fashion, that is, content messages are first aggregated into sentences conveying intra-sentential rhetorical relations, and then the inter-sentential relations are established.

Within-sentence structuring is performed as follows. Given a list of (filtered) content messages produced in the previous Content Determination stage, we would like to have them distributed across a number of individual sentences. To this end, we defined training instances of within-sentence structuring as relations between message pairs in the form (m_1, m_2, *relation*) in which m_1 and m_2 are messages represented as attribute-value pairs, and *relation* is a rhetorical relation (which in our data could be either *concession*, *joint* or *contrast*). For each positive instance of within-sentences structuring (m_i,m_j), we have also defined counter-examples (conveying the '*none*' value of the *relation* class) covering every other possible message combination. Thus, our goal was to use messages as learning features for classifying *relation* as one of its possible RST values, or as the special *none* case. 12,247 such training instances of within-sentence structuring were extracted from the *SINotas* corpus, being 394 two-message sentences (in which messages are linked by a RST relation) and the reminder 11,853 instances being one-message (*none*) sentences as follows:

$[attr_1, val_1, attr_2, val_2, relation]$

Between-sentences structuring follows a similar approach. Our goal in this case is to learn possible rhetorical relations between the sentences produced in the previous stage (which in our data could be either *contrast, elaboration* or *none*.) Thus, every related sentence pair in the corpus produced a positive training instance in the form (m_1, m_2, *relation*) in which m_1 and m_2 are messages from one sentence each, and which are found in the nucleus and satellite (or vice-versa) of a inter-sentential rhetorical relation.

For each positive instance of between-sentences structuring (m_i,m_j), we have also defined 12 negative instances (conveying the '*none*' value of the *relation* class) covering every other possible message combination (m_i,m_k)such that $j \neq i$ and $j \neq k$. In this way, 3432 training instances were extracted from the corpus,

being 176 cases of *contrast*, 88 cases of *elaboration* and 3168 counter-examples (*none*). The structure of the training instances is the same used for within-sentence structuring, though considering inter-sentential RST relations:

$$[attr_1, val_1, attr_2, val_2, relation]$$

4. Implementation

The message generator was implemented as a module that produces a 14-message vector from the given set of input values (e.g., students' grades etc.) Similarly, the message selector was implemented as a subsequent module that reduces this vector to those (possible fewer) messages that should actually be passed on to the next stage. Both modules were integrated (or rather, pipelined) as a Content Determination component corresponding to the first stage in our NLG application under development.

Following the same approach, within-sentence and between-sentence structuring were pipelined in a Document Structuring module. In this case however it was necessary an additional procedure for submitting all possible message pair combinations to each classifier in order to decide which message pairs should make sentences and which sentences should be linked by rhetorical relations.

A complete example of our classification-driven Document Planning works as follows. First, Content Determination: given a student's record showing (among other information) a 6.6 grade in the final exams, data interpretation produces a fixed 14-message vector conveying all relevant facts about the input, including the message *mf_aval=bom* (which stands for a 'good' grade in the final exams.) Next, content selection takes this vector as an input, eliminates all unnecessary messages and outputs the (sub)set of those that should actually appear in the text.

The second stage is Document Structuring: within-sentence structuring could determine that the generated message should be aggregated in a single sentence with, say, a message *mf_turma=abaixo* (which says that the grade falls below the overall class results) using a *concession* rhetorical relation. Finally, between-sentences structuring could link the generated sentence to a second one using a *contrast* rhetorical relation. In this example, the single piece of information about the final exams (had we implemented the entire system, of course) could eventually be realised as "Your grades in the final exams were good but below average", and then linked to another sentence as "On the other hand, your substitutive examination grades were pretty good".

5. Results

To each of our four Document Planning subtasks – data interpretation, content selection, within-sentence and between-sentences structuring - we have applied J48 Weka [14] decision-tree induction using 10-fold cross-validation and its default parameter values.

With respect to data interpretation, we notice that each message actually depends only on a small set of features. This does not come as a surprise as our 25 learning features cover a wide range of phenomena in the application semantics, e.g., from weekly attendance to average grades. Thus, all decision-trees were revised to determine which learning features were actually needed for each classification, and pruned accordingly. As a result, the 14 classification tasks were performed using on average only 2.2 learning features each.

Four messages types (*sub_aval*, *sub_turma*, *corel_nota_falta* and *aband_rec*) could not be classified automatically due to the heavy imbalance in their value distributions and/or data sparseness. These cases will be implemented separately following a knowledge-engineered approach. Table 1 below shows the results for the reminder (i.e., machine-learned) classes only as compared to a baseline approach that simply selects the most frequent value for each class.

Table 1. Data interpretation results

	Decision-tree induction			Baseline
Class	Prec.	Recall	F-measure	Correctness
provas_aval	0.970	0.968	0.982	0.349
provas_turma	0.989	0.989	0.989	0.415
progresso	0.756	0.754	0.752	0.270
eps_aval	0.710	0.635	0.659	0.502
dev_ep1	0.963	0.963	0.963	0.859
freq_aval	0.918	0.948	0.945	0.780
mf_aval	0.956	0.958	0.977	0.336
mf_turma	0.988	0.984	0.986	0.560
rec_aval	0.928	0.931	0.925	0.830
rec_turma	0.652	0.716	0.680	0.846

A number of observations are due. First, the frequency-based baseline would only approach decision-tree induction when data are extremely sparse (e.g., the class *rec_turma* in which 85% of values are '*null*'.) On the other hand, when these problems do not occur (e.g., the values of the class *mf_aval* are evenly distributed in the corpus) results of the machine-learned approach are far superior to the baseline.

Second, we notice that accuracy rates for some classes are extremely high, which is mainly the case of classes of well-defined semantics (for example, University policies determine that a 5.0 grade should be considered average, and this kind of knowledge was taken into account by the domain expert when writing the reports.) However, this is not the case when we consider more subjective classes such as, e.g., *progresso*, which describes the student's performance curve throughout the term (e.g., rising, falling, U-shaped etc.), or sparse data such as in *rec_turma* (modelling recuperation exams attended by very few of the students.)

In addition to that, we notice that the results hide some extreme situations in all classes. For example, the data do not provide sufficient evidence to classify instances conveying *rec_turma=media* (i.e., an average result in the recuperation exams) which occurred only twice in the training data. In practice, this means that we

should expect the system to rate an 'average' grade as something else in 2 out of 241 cases (0.83%), a relatively minor error rate that we expect to accommodate in the subsequent (and also classification-driven) stages of the generation process. In either case, we believe that the overall positive results in message classification should be interpreted as more indicative of the simplicity of the underlying semantics (allowing the authoring of highly consistent reports as seen in the corpus) and less of the performance of the annotation task or computational approach undertaken.

With respect to the second task - content selection – we notice that some of the message realisations turned out to be trivially derivable from the input messages. This was the case of five messages conveying highly prominent information (*mf_turma*, *corel_nota_falta*, *provas_turma*, *dev_ep1* and *abandon_rec*) that is always included in the output text unless they contained a '*null*' value. For all these cases, no learning approach was actually required. Below we show the results for the reminder (machine-learned) classes and the correctness rates of a possible baseline algorithm that simply includes all non-null messages in the output text.

Table 2. Content selection results

	Decision-tree induction			Baseline
Class	Prec.	Recall	F-measure	Correctness
provas_aval	0.995	0.981	0.987	0.784
sub_aval	0.995	0.944	0.968	1.000
sub_turma	0.875	0.991	0.924	0.938
progresso	0.989	0.963	0.975	0.983
eps_aval	0.894	0.916	0.904	0.784
freq_aval	0.884	0.884	0.884	0.817
mf_aval	0.821	0.923	0.849	0.784
rec_aval	0.963	0.992	0.977	0.988
rec_turma	0.993	0.937	0.963	0.946

Although content selection turned out to be a straightforward procedure – the simple baseline algorithm achieves superior results in 4 out of 9 classes - we notice that by applying the generated models to the 241 input message vectors in the *SINotas* corpus we have actually obtained output sets of average 5.1 messages each[7], that is, the next module to be implemented – document structuring – will have to deal with an average of 5 messages in each text, and not 14, a considerable reduction in the task complexity that we expect to become evident in the next stages of development of our generation system.

Results for the third task - within-sentence structuring – are as follows.

[7] This relatively low average number of output messages is greatly influenced by several records of students that did not sit any of the expected exams, producing single-sentence reports of the kind "Unfortunately you do not seem to have followed the course regularly".

Table 3. Within-sentence structuring results

	Decision-tree induction		
Class	Precision	Recall	F-measure
concession	0.980	0.916	0.947
joint	0.847	0.756	0.799
contrast	0	0	0
null	0.993	0.997	0.995

The high correctness rates in this case are due to the regularity of the sentence structures in the corpus, most of which either comparing two opposite values (e.g., a low and a high grade in a *concession* relation) or simply stating them as two otherwise independent facts (linked by a *joint* relation.) The corpus contained only eight instances of intra-sentential *contrast* relations. These cases could not be classified automatically, and will be left out from our output documents. Given the high F-measure rates above, we do not explicitly provide a baseline algorithm. By comparison to this case we notice that the simple choice for, e.g., the most frequent meaningful class (*concession*) would have achieved 70.8% correctness rate, but only if we could disregard the negative instances, that is, if the system somehow 'knew' in advance that the two messages should be linked by a RST relation in the first place.

Finally, results for the fourth task - between-sentences structuring - are as follows.

Table 4. Between-sentences structuring results

	Decision-tree induction		
Class	Precision	Recall	F-measure
contrast	0.933	0.875	0.903
elaboration	0.882	0.852	0.867
null	0.990	0.995	0.992

Once again, we observe high correctness rates due to the uniform rhetorical structure of our corpus. By means of comparison, a possible baseline strategy that chooses always the '*null*' class would achieve up to 92.3% correctness (although obviously not performing any useful document structuring.)

6. Final Remarks

We have presented a corpus-based approach to NLG Document Planning addressing the initial stages of Content Determination (here including both aspects of data interpretation and content selection) and Document Structuring (which we called within- and between-sentences structuring.) Although still domain-dependent - in the sense that it requires annotated training data – results in both cases were highly satisfactory, suggesting that the general methodology is in principle applicable to the development of simple NLG systems of this kind. Moreover, as automatic (or semi-automatic) methods for data-text alignment become more widespread (e.g., [12]), the current knowledge acquisition bottleneck is likely to become more treatable.

We are currently working on the late stages of Macroplanning / Microplanning, that is, generating abstract sentence specifications for a future surface

realisation module, which should ultimately lead to a simple NLG system made of a series of classifiers.

7. Acknowledgements

The authors acknowledge support by CNPq, FAPESP and the University of São Paulo (USP).

8. References

[1] Reiter, E. and R. Dale “Building Applied Natural Language Generation Systems”. Cambridge University Press, 2000.

[2] Reiter, E. “An Architecture for Data-to-Text Systems”. In Proceedings of ENLG-2007, pages 97-104, 2007.

[3] Mellish, C. et. al. “A Reference Architecture for Natural Language Generation Systems”. Natural Language Engineering 12 (1) pages 1–34, 2006.

[4] Mann, W. C. and S. A. Thompson “Rhetorical Structure Theory: A Theory of Text Organisation”. L. Polanyi (ed.) The Structure of Discourse. Ablex, Norwood, 1987.

[5] Portet, F., E. Reiter, A. Gatt, J. Hunter, S. Sripada, Y. Freer, C. Sykes "Automatic Generation of Textual Summaries from Neonatal Intensive Care Data". Artificial Intelligence 173, pages 789-816, 2009.

[6] Williams, S. and Ehud Reiter “Deriving content selection rules from a corpus of non-naturally occurring documents for a novel NLG application”. Corpus Linguistics workshop on Using Corpora for Natural Language Generation, 2005.

[7] Geldof, S. “Corpus analysis for NLG”. In: Reiter, E., Horacek, H. and van Deemter, K. (Eds.) 9th European Workshop on NLG, pages 31-38, 2003.

[8] Sripada, S., E. Reiter, J. Hunter, and J. Yu “Exploiting a parallel text-data corpus”. Corpus Linguistics 2003, pages 734–743, 2003.

[9] Marciniak, T. and M. Strube “Using an Annotated Corpus As a Knowledge Source For Language Generation”. Corpus Linguistics’05 Workshop Using Corpora for NLG (UNNLG-2005), pages 19-24, 2005.

[10] Smets, Martine, Michael Gamon, Simon Corston-Oliver and Eric Ringger “French Amalgam: A machine-learned sentence realization system”. TALN-2003, Batz-sur-Mer, 2003.

[11] Duboue, Pablo A. and Kathleen R. McKeown. "Statistical Acquisition of Content Selection Rules for Natural Language Generation". EMNLP ‘03, pages 121–128, 2003.

[12] Barzilay, Regina and Mirella Lapata. "Collective Content Selection for Concept-To-Text Generation". In HLT‘05, pages 331–338, 2003.

[13] Kelly, Colin, Ann Copestake and Nikiforos Karamanis “Investigating Content Selection for Language Generation using Machine Learning”. 12th European Workshop on Natural Language Generation, Athens, Greece, 2009.

[14] Witten, I. H. and E. Frank. “Data Mining: Practical machine learning tools and techniques”. 2nd edition, Morgan Kaufmann, San Francisco, 2005.