Evolving Questions in Text Planning

Mick O'Donnell
Escuela Politecnica Superior,
Universidad Autónoma de Madrid,
Cantoblanco, Spain
michael.odonnell@uam.es

Abstract

This paper explores how the main question addressed in Text Planning has evolved over the last twenty years. Earlier approaches to text planning asked the question: *How do we write a good text*?, and whatever answers were found were programmed directly into code. With the introduction of search-based text planning in recent years, the focus shifted to the evaluation function, and thus the question became: *How do we tell if a text is good?* This paper will explore these evolving questions, and subsequent refinements of them as the field matures.

Introduction

Given the growing interest in the application of search-based planning methods in NLG, we can describe how text planning has evolved from its beginning in terms of the kinds of questions that have been addressed.

1 How do we compose a good text?

The object in text planning is to write a program which can compose a text to meet some input specification. Traditionally, the programs themselves embody decision making procedures coded to make textual decisions, such as whether or not to include some content, how to rhetorically link it to the rest of the text, and in what order should the content nodes appear.

How one writes a good text has many answers, and depends partly on the type of text one is writing. For some texts, a schematic approach is possible: a text can be composed by using predefined schemas, which nominate what information should appear, and in what order (McKeown 1985). Sometimes the system makes the inclusion of a schema element conditional on the context of the text, for instance, deciding to include or exclude content based on the type of user that the message is being generated for (DiMarco *et al.* 1997).

For other types of text, the application of schemas is less appropriate. Some text planners take as their goal the delivery of a single fact, or a set of facts, but take into account that other information may need to be given first for each message to be understood, other information may need to be given after to counter misconceptions, and examples may be given to aide assimilation (for instance, the RST text planners, such as Hovy 1988). The system thus composes a text which delivers the facts required of it, and any other facts which will facilitate the uptake of these facts.

In another text genre, the goal is not to deliver specific facts, but rather to describe an entity, for instance, museum artefacts (e.g., ILEX: O'Donnell *et al.* 2001), or animals (*Peba*: Milosavljevic 1999). A basic strategy to generate a reasonable text of this kind is to list the different attributes of the target entity, possibly sorted into relevant topics. Good systems are aware of which information is already known to the intended audience, what should be interesting, etc. In ILEX, the system followed various strategies to generate better texts, such as allowing digressions (describing secondary entities), generalisations, defeating misconceptions, aggregation, etc.

Systems have increasingly tried to address larger sets of the issues needed to produce good quality text. We not only need to ensure that appropriate content is selected, and that adequate grounding is given to ensure the audience can understand the content. Increasingly issues of coherence dominate. Entities should be referred to in a contextually appropriate form, related sentences should be aggregated to avoid repetition, the sequence of sentence themes should be structured to express a logical development of messages, etc.

As systems address more and more of these issues, the planning algorithms become more complex. Systems which need to produce texts meeting content and coherence constraints are reaching the limit of what they can do, because the programming needs to address too many issues.

2 What texts can be produced?

The text-planning problem gets more complex when text planners need to address global as well as local goals. In Ilex, one constraint on the planner was that the generated text needed to fit in a specified space on a web page. During the generation process, the decision of whether or not to include a given fact in the page was difficult to make, since while the inclusion of the fact might help achieve the communicative goals, it also increases the size of the text. However, the system cannot know until the whole text is produced whether including this fact would push the text size over the limit. One can only calculate the real cost of including extra content when the text is finished. In the Ilex case, we selected an incremental text planning approach, adding facts into the text structure in order of their relevance, until the space limit is reached. But this approach is not compatible with all generation goals.

In the 1990s, several approaches started to recharacterise the text planning process as one of searching the space of possible text structures for those structures which were optimal when seen as a completed whole. These approaches split the question: *how do we produce a good text?* into two parts:

- How can we search the space of texts which could be produced for given content?
- How can we select the text in this space which optimally meets our goals?

In this section, we will discuss the first of these questions. Marcu (1997) characterised the text planning process as follows. He assumed that the knowledge base (KB) consists of a set of semantic units to express, and that an additional resource provided a set of rhetorical relations, which can hold between two semantic units in the KB. A pair of semantic units may have no relation between them, one relation, or multiple.

The set of units to express is finite, and the relations that can hold between them is finite, so if we assume that each semantic unit can only appear once in the text tree, then it is clear that the set of text structures that can realise the KB is also finite. We can model the space of possible text structures as a lattice, where each point in the lattice represents a given set of included facts, with a given set of relationships between them, and a given sequencing of these facts in relation to each other.

Seen in this light, one might naively propose the following approach to text planning: generate all possible text plans for the given input, evaluate each tree as a whole, and choose the one with the highest value. However, even for a small number of facts, the number of trees is huge, and this approach is not feasible.

A more feasible approach is to treat this as a search problem. Each text plan represents one point in the space of all possible plans. Adjacent points in this lattice represent similar text plans differing in one detail (the insertion of a fact, the deletion of a subtree, the reordering of nucleus and satellite, etc.). Given a starting text plan, we can move from point to point in the space, searching for a text plan which is optimal for our goals.

The earliest of work to take this approach was that of Zukerman and McConachy (1994). They were concerned with generating texts which conveyed content adequately but concisely. A text adequately conveys the facts it is given to convey if these facts are understood by the addressee. Where a given fact requires other facts to be given for it to be understood, these facts need also to be included. If a given fact or set of facts may cause the reader to make an unintended inference, facts are included to prevent this. Additionally, the inclusion of a fact sometimes made the inclusion of other facts unnecessary (e.g., including an example might clarify a concept better than a longer explanation). The conciseness goal required that as few facts as possible are used to convey the information that needs to be conveyed.

Given these constraints, traditional planning methods which made decisions locally were not appropriate. The conciseness and communicative adequacy of the text is a property of a set of facts as a whole, and thus text plans need to be evaluated as a whole. They thus rephrased the text planning issue as optimisation at a global text level.

They only addressed part of text planning as an optimisation problem: content selection. They take as a starting point the set of facts that are categorically required to be expressed. They then use an incremental graph search algorithm to search for optimal sets of facts to include in a text: on each iteration, facts are added to or deleted from each solution to produce new solutions, until the program finds one solution which adequately conveys the designated facts, with the fewest facts possible. Marcu (1997) experimented with several different search methods. Firstly, he assumed that *all* the units in the KB need to be included in the text plan (the problem of content selection was thus not relevant). His goal was thus to find RST trees which optimally covered all facts.

In his first algorithm, he used a chart mechanism to construct all possible text plans for the given content and the set of possible relations. However, as he notes, for any non-trivial number of facts, with multiple possible relations between them, this approach is computationally expensive. I also note that Marcu assumed all facts in the KB were to be expressed. If content selection is included in the process, meaning all facts are optional, the solution is even less tractable.

His second proposal restricts the chart mechanism to a greedy algorithm, which at each level, selects only the best tree to take to the next level. However, this basically amounts to make decisions locally, not globally, and thus is not relevant to this section.

His third algorithm is more relevant here. He splits the problem into two parts. Firstly, determine an ordering of content units such that as many as possible of the ordering and adjacency constraints in the potential relations are satisfied. Secondly, derive an RST tree for this ordering of content units.

He uses a constraint satisfaction algorithm for the first step. A small corpus study was used to derive, for each relation, the relative probability of nucleus-satellite vs. satellite-nucleus ordering, and also the degree to which the connected text segments are required to be linearly adjacent.

Given a set of facts, for each possible relation between a pair of facts, the process asserts a constraint stating what the nucleus-satellite order should be, and another constraint asserting that the connected facts should be adjacent. The strength of the constraint depends on the values obtained in the corpus study, for instance, if a relation allows relatively free ordering of a satellite, the constraint strength would be low.

The constraint satisfaction algorithm then derives the ordering of facts which maximises the constraints. Marcu elsewhere provides an algorithm to build a text tree on top of a given ordering of facts, again using adjacency and order propensities.

Mellish *et al.* (1998) made the point that even this restricted approach would soon become intractable with more than a small set of facts when one allows weak RST relations such as Joint and Elaboration into the model.

An alternative approach, which I haven't seen implemented (although Mellish mentions the possibility), would be to apply hill climbing techniques to the problem of finding a text which optimally satisfies a set of locally and globally stated text constraints. The idea would be to start with a single text structure, generated in a simple manner. The system then tries each possible single mutation on this text structure, (e.g., adding a fact in each location in the tree this is possible; deleting each subtree in the tree; grafting a subtree from one location to another, etc.). Each of the resulting text structures is then evaluated, and the one which scores higher is taken as the text structure for the next cycle. When no one-step change to the tree results in an improvement, the process stops.

The problem with all hill-climbing applications is that we might reach a local maximum, a point where no simple change can produce an improvement, but radical restructuring could produce better structures. One partial solution is to repeat the process a number of times from different starting structures, and select the best structure produced in all the trials. But the problem remains.

Genetic algorithms offer an alternative to hill-climbing methods, and are less susceptible to stopping at local maxima. In the late 1990s, Chris Mellish implemented the first stochastic text planner (Mellish *et al.* 1998). He replaced the procedural text planner of the ILEX system, with one based on a genetic approach.

Like the above approaches, a genetic algorithm can be seen as a means to search the space of possible texts. The system starts with a small population of randomly produced text structures, and, on each iteration (each generation), randomly mutates some or all of the structures. Each new text structure is then evaluated by an "evaluation function".

In each generation, those text structures with the highest 'fitness' are more likely to produce offspring, while those under a certain threshold are dropped from the population. In this way, mutations which improve the text structure should be preserved in the population, while those which weaken the text should disappear. After a certain number of iterations, the process is stopped, and the highest scoring text-structure is selected as the one to give the user.

The advantage of a genetic algorithm over a hill-climbing approach is that mutations which may by themselves lower the text value can survive in the population, and later combine with other mutations to produce a high-value text structure. The evolutionary approach has its cost, however, in that far more processing time is required. For this reason, the text mutation stage is usually kept fairly simple.

Like Marcu, Mellish also assumes that the search algorithm is given a set of facts, all of which need to be included. He proposes 3 algorithms. The first mutates a selected tree by randomly swapping two subtrees, and then 'repairing' any relation which is no longer valid (where the relation which links a nucleus and satellite no longer holds between these facts, select another relation which does, or if none exist, use the generic relation, Joint).

Mellish noted that, often, different trees in the population develop good structure for different subsets of facts. He thus introduced a cross-over mechanism, allowing the grafting of structure from one tree to another. He simplifies the process somewhat in a manner similar to Marcu, assuming that the genetic algorithm only manipulates the surface ordering of facts, and that an RST tree can be derived from that ordering (he assumes satellites always follow the nucleus). He differs from Marcu in that evaluation is of the final text tree, rather than the fact sequence. Mellish thus allows for two forms of mutation: moving a fact from one place to another within a single fact sequence, and inserting a sequence of facts from one fact-sequence into a random place in another (then deleting any facts which are repeated).

I have two problems with the approach above. Firstly, I do not believe that a given sequencing of facts so deterministically relates to a particular tree structuring of those facts. Two trees with the same surface ordering of facts could be fundamentally different in structure and in terms of the relations used, and thus will be evaluated very differently. By not including the text structure in the search process, the process cannot search for an optimal text structure.

My second problem with this approach is that the mutations used are fairly destructive of the text plans. When as a human writer I cut and paste text from one document to another, I generally find the amount of repair needed out-weighs the time saved by reusing text. And even the simpler mutation, randomly moving a text node within a sequence, will have far more chance of fracturing coherence than of creating it.

For these two reasons, I believe it is better to use a genetic text planner which firstly operates directly on text trees, and secondly, only allows mutations which preserve the legality of the tree. I also assume the genetic algorithm is used to determine selected content as well as structuring and ordering it. In this hypothetical text planner, no cross-over is used, only operations on a target tree. We start with a population of trees of one fact each (all related to the entity being described). The mutations include:

- *Insert Fact*: select a fact not in the tree and attach it as satellite to a fact which permits such a relation.
- *Delete Subtree*: randomly select a subtree and delete it.
- *Move Subtree*: randomly select a subtree and move it to another location where it can legally attach.
- *Change Top Nucleus*: break off a subtree from the main tree, make its nucleus the top of the tree, and graft the original tree into this tree at a legal point.
- *Switch Order*: change the order of a satellite in relation to the nucleus, or in regards to other satellites of the same nucleus.
- *Switch Focus*: the assessment of text trees includes the evaluation of the focus movement throughout the tree, each fact in the tree nominates a focus. This operation changes that focus for another of the entities in the fact, which will affect which entity is realised as subject in the final text.

All of these operations preserve the legality of the tree, but may change the global evaluation of the tree, given that suboptimal focus movements may result, information prerequisite for understanding may occur late, or not at all, etc.

By only allowing legal mutations, the system needs to do less work repairing the illegalities introduced by mutations. Good solutions should be reached quicker, although as said above, this approach is still in an early stage of development,

To a degree, this approach mirrors the way humans write texts. We start off with a rough draft with the core of what we want to say, and revise parts, sometimes adding in an explanation, sometimes adding a digression to provide background material. We may erase material because a change in another part of the document made the material redundant. Or we might cut/paste material from one part of the document to another.

Another approach to the problem of satisfying global constraints in text planning has been the 'generate and revise' approach, where a text is generated with only partial regard to the global constraints, and the resulting text is then revised to fit the global constraints (e.g., Robin and McKeown, 1996). In the STOP system (Reiter 2000), texts are constrained to fit a certain length. The system generates texts of approximately the right length, and then prunes the text until the size requirement is met. Piwek and van Deemter (2007) expand this approach to handle more than a single global constraint, exemplifying using both global length and communicative effectiveness as (sometimes) contradictory goals. Their approach is to generate a single starting text, and then apply revision operations to work towards an optimal text.

If one sees a revision operation as similar to the text mutations applied above, then it seems the revision approach is not so different to the genetic algorithm approach above. One difference is that the revision approach uses procedural means to generate a reasonable starting text, while the stochastic approaches generally start with easy-to-generate texts and mutate these towards higher complexity and optimality. Another important difference, at least in the work of Piwek and van Deemter, is that they apply their revisions exhaustively, looking at all possible combinations of revisions to locate the optimal one. For more complex text planning tasks, this may become less tractable.

3 How do we tell if a text is good?

One of the consequences of using genetic algorithms for text planning is that much of one's effort is spent on developing the evaluation function, defining the formulas used to decide how good a particular text is. The main focus in building a text planner thus moves from deciding *how do we compose the text?* to one of deciding: *how do we tell if a text is good?*

The important point here is that with procedural planners, we need to take a cognitive approach, trying to perceive how a text is constructed from scratch, which decisions are made, and in what order. Using an evolutionary approach, we can ignore the *process* of writing, and focus on the *products* of writing. We can then function as linguists: our concern is to decide what makes a text function well, and what interferes with its success. These are more important, more abstract questions to address.

Of the systems discussed above, the evaluation functions vary. The system of Zukerman and McConachy (1993) introduced the idea of global criteria for evaluating text, such as conciseness (meeting the informational goals of the system in the fewest facts), and ILEX used text length as a global criteria. Marcu (1997) set the goal of maximising 'global coherence' of the text, which basically amounts to maximising the sum of local decisions made throughout the tree. Marcu used a corpus study to determine how flexible the adjacency and ordering constraints of each relation were, and penalised instances of the relation which did not meet the constraint, the size of the penalty depending on the observation in the corpus.

He does however mention that his approach could be set up to "ensure that the resulting plans satisfy multiple high-level communicative goals" (p629).

The evaluation function of Mellish *et al.* (1998) also was calculated over a sum of local features of the tree, although a wider set of features were involved. These included: rewarding 'interesting' relations and penalising the Joint relation; penalising separation between a nucleus and satellite; penalising unsatisfied precondition of a relation; rewarding good focus movements and penalising bad ones.

While it is clear each of these criteria is contributory to good text, the numbers used were made up. Basing these numbers on a corpus study of good texts may be useful for future work.

Following on from Mellish *et al.*, some work took place focusing on improving the evaluation function, for instance Cheng and Mellish (2000) expanded on the criteria for evaluating focus movement (there called 'discourse topic movement'), and also allowed for embedding of facts in others, providing criteria for evaluating how good an embedding is. Karamanis also examined focus movement (or as he calls it, *entity coherence*) in text evaluation (Karamanis and Manurung 2002; Karamanis 2003; Karamanis *et al.* 2004; Karamanis and Mellish 2005). He proposes dropping the use of rhetorical structure, and taking the goal of text planning as sequencing facts so as to achieve smooth focus movement. Mutations thus change the ordering of facts in a given sequence (as in the Mellish work), but evaluation is applied directly to the fact sequence, penalising discontinuous focus movements.

In summary of this section, the movement to search-based text planning allowed the researcher to move away from issues regarding how to program a system to generate texts, focusing instead on the problems of deciding how to evaluate the quality of a text.

4 How good is this text?

A major problem with the search-based systems described above is that the system-maker needs to formulate the evaluation function: to specify the criteria used to determine how good a text is. This is a difficult problem, with no simple answers.

In some cases, corpus studies have been performed to find the patterns used in good texts, and base the evaluation metric on these results. However, this approach requires significant amounts of time spent on corpus analysis.

One way around this problem is the use of machine learning (ML). The basic idea in ML is to provide the computer with lots of example texts, along with the classification of the text. The system then builds a classification model from the training data. This classification model can then be applied to previously unseen examples to assign a class.

Applying ML to text planning, we provide the computer with a corpus of finished texts, each rated from 0 to 10 in terms of quality, and leave the computer to work out an evaluation function on its own. This step would move the work of the investigator from deciding how to evaluate a text to a far simpler one, of just saying *if* each text is good or not, something we all can do with or without linguistic training.

The problem here is that ML techniques can only function on the range of features that they have access to. In the worst case, only the surface text is provided to the system. Some approaches work simply with the n-gram sequences of words, for instance, to select between alternative sentences generated to express some content (e.g., Langkilde and Knight 1998). N-grams have even been used to assess the global quality of texts, not for text planning, but to assess student exam questions (Pérez *et al.* 2004). However, the sequence of words within a sentence cannot tell us anything about the quality of the text structure.

In recent years, there has been substantial work that assumes that the quality of a text can be judged in relation to the degree to which the sequence of sentences in the generated text corresponds to a 'golden standard' (Lapata 2003; Dimitromanolaki and Androutsopoulos 2003; Karamanis *et al.* 2004). This work, referred to as 'Input Ordering', assumes that content selection is done as a separate task, and the role of text planning can be seen as simply ordering the facts output from content planning. A corpus of human-written texts, (in most approaches tagged by propositional content), is provided. These represent a "golden standard". Text plans generated by the computer are evaluated in terms of the degree to which the terminals of the text structures occur in the same order as the golden standard.

For me, this approach is problematic: two texts may have the same surface ordering of facts, yet very different rhetorical structure. One text may be well structured, and the other badly structured. The surface ordering of facts is by itself a poor indicator of the overall quality of the text.

This recent focus on input ordering results partly from the fact that it is an aspect of text that can easily be recognised. Focusing on input order just because it is easy is something like looking for your watch under the street-light, even though you lost it in the dark alley. The looking might be easier, but if the answers are in the dark, then that is where we should be looking.

Rather than explore surface features of language which are easier to recognise, I believe we should be either:

- building discourse-level tree-banks of real texts, to provide real information to inform the automatic derivation of evaluation functions, or,
- building tools to automatically recognise discourse structure of text, (RST, focus movement, etc.)

Daniel Marcu has been a leader in both directions, working both on building an RST-tagged corpus, and also exploring the automatic recognition of rhetorical structure (Marcu 2000). In regards to the latter, unfortunately, the results have not so far been useful for real applications. If one wants to rhetorically annotate a corpus, one needs to do it by hand, which involves a substantial investment of time. The case is similar for annotation of other discourse structures, such as focus movement.

Within the context of NLG, another approach is possible. For each generated text, we have at hand the deeper structure which was produced in constructing the text. In the case of ILEX for instance, a side-product of generating a text is the rhetorical structure of the text, information regarding co-identity of the entities mentioned in the text, etc. The machine-learning program can thus be provided with all of this information as input, as well as derived focus-movement, etc. We can produce a corpus of generated texts, tagged with structural information, and ask a human, to assign a level of goodness to each text. The program can then derive an evaluation function of its own.

The work of the human in the process thus becomes simply to assign a level of goodness assessment to each of the texts in the training set.

5 What features of a text are critical in evaluating worth?

The effectiveness of any machine leaning program is very limited by the value of the text features it receives. While the previous point started out with a focal question *How good is this text?*, we saw that the real question becomes: *What factors do we feed the system to optimise classification?*
In a sense, this is a movement back towards question 3: *How do we tell if a text is good?* However, note the difference: question 3 concerns the investigator assigning values to particular text features, typically making them up as they go.

With this new question however, the investigator is not *assigning values* to text features. Rather, the investigator need only include the feature in the data, and it is up to the learning algorithm to decide to what degree that feature improves or worsens the text quality. The investigator only needs to hypothesise whether the text feature could be influential in this decision.

The main advantage here is the movement away from investigators making up evaluation function in their heads, or performing corpus studies. In the machine learning approach, we need only include the feature in the corpus. For instance, the evaluation function of Mellish *et al.* (1998) assigned +3 for each instance of subject-repetition. In an ML system, the human just decides that subject-repetition should be considered, and the system derives an appropriate parameter for this feature.

It is possible that many of the features we provide to the system are not actually used by the system. If we use an ML approach which allows access to its evaluation model in an understandable form, then the approach has the added value of revealing to the investigator *which* of the features provided to the system are important to text value, and which are not. One product of this approach, apart from an improved text generator, will be an improved understanding of what it is that makes a text good or bad.

Early approaches to applying ML in this way are calculating the relative frequency of particular discourse features (e.g., of types of focus movements, of types of rhetorical relations and their ordering). Each generated text can then be evaluated in terms of the relative frequency of its discourse features in comparison with the tagged corpus.

One problem with this approach is that many aspects of text structure cannot be said to be good or bad *per se*, it is only in relation to the context of use that such can be judged. Almost any RST relation can appear in a text, and the value of the use needs to be judged in the context of its appearance. However, it is difficult for a machine learning system to infer in which contexts the use of a particular relation is good and in which it is bad.

We have assumed a supervised learning approach, where a human evaluated the quality of each text in the corpus. Because there is a cost to producing such a corpus (human evaluation of each text), some prefer an *unsupervised* approach, whereby the input at the training phase is not valued: it is assumed that all of the texts in the training corpus are good representatives of the genre.

6 Summary

We have represented recent developments in text planning in terms of an evolution in the principle question addressed. While earlier work in text planning addressed the question of *how we compose texts*, the movement into search-based planning split the question into two parts: *how we search the space of possible texts*, and *how we evaluate a text*.

The second of these questions has been increasingly in focus, and the need to move away from made-up evaluation functions leads to the corpus-based derivation of text metrics. At first glance, this seems to leave the analyst with the simple task of evaluating the overall quality of a text, and leaving it to the machine to derive the evaluation function (the question thus becomes: *how good is this text?*).

However, on deeper analysis, the learning system needs to be fed structured input to be able to derive intelligent evaluation functions. The focal question thus needs to shift to considering *what features of discourse structure need to be provided to the ML system.*

The field of search-based text planning is still young, and evolving quickly. In the last 10 years, the focal questions have evolved rapidly. The question is, what will we be asking in another 10 years?

References

H. Cheng and C. Mellish. 2000. Capturing the Interaction between Aggregation and Text Planning in Two Generation Systems. *Proceedings of the 1st Int. NLG Conference*, Mitzpe Ramon, Israel, 108-115.

A. Dimitromanolaki and I. Androutsopoulos. 2003. Learning to order facts for discourse planning in natural language generation. *Proceedings of the 9th European Workshop on NLG.*

C. DiMarco, G. Hirst and E. Hovy. 1997. Generation by selection and repair as a method for adapting text for the individual reader. *Proceedings Workshop on Flexible Hypertext*, 8th ACM International Hypertext Conference, Southampton, U.K., April 1997, 20-23.

E. Hovy. 1988. Planning coherent multisentential text. *Proceedings of the 26th Annual Meeting of ACL*, 163-169.

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

N. Karamanis and H. Manurung. 2002. Stochastic Text Structuring using the Principle of Continuity. *Proceedings of the 2nd Int. NLG Conference.*

N. Karamanis, C. Mellish, J. Oberlander and M. Poesio. 2004. A corpus-based methodology for evaluating metrics of coherence for text structuring. *Proceedings of INLG04*, Brockenhurst, 90-99.

I. Langkilde and K. Knight. 1998. The Practical Value of N-Grams in Generation. *Proceedings of the 9th Int. Workshop on NLG*, Ontario, Canada, 1998.

M. Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. *Proceedings of ACL 2003*, 545-552.

D. Marcu. 1997. From local to global coherence: A bottom-up approach to text planning. *Proceedings of the National Conference on Artificial Intelligence* (AAAI'97), 629-635.

D. Marcu. 2000. The Rhetorical Parsing of Unrestricted Texts: a Surface-Based Approach. *Computational Linguistics*, 26 (3), 395-448.

K. McKeown. 1985. *Text Generation*. Cambridge University Press, Cambridge.

C. Mellish, A. Knott, J. Oberlander and M. O'Donnell. 1998. Experiments using stochastic search for text planning. *Proceedings of the 9th Int. Workshop on NLG*, Ontario, Canada, 98-107.

M. Milosavljevic. 1999. *The automatic generation of comparisons in descriptions of entities*. PhD Thesis. Department of Computing, Macquarie University, Australia.

M. O'Donnell, C. Mellish, J. Oberlander and A. Knott. 2001. ILEX: An architecture for a dynamic hypertext generation system. *Natural Language Engineering* 7, 225-250.

D. Pérez, E. Alfonseca, and P. Rodríguez. 2004. Application of the BLEU method for evaluating free-text answers in an e-learning environment. *Proceedings of the Language Resources and Evaluation Conference* (LREC-2004), Portugal.

P. Piwek and K. van Deemter. 2007. Generating under Global Constraints: the Case of Scripted Dialogue. *Journal of Research on Language and Computation*, 5(2):237-263.

E. Reiter. (2000). Pipelines and Size Constraints. *Computational Linguistics*. 26:251-259.

J. Robin and K. McKeown. 1996. Empirically designing and evaluating a new revision-based model for summary generation', *Artificial Intelligence*. 85(1-2).

I. Zukerman and R. McConachy. 1994. Discourse Planning as an Optimization Process. *Proceedings of the 7th Int. Workshop on NLG*, Kennebunkport, Maine, 37-44.