Engineering Text Generation

Alain Polguère
Département de linguistique et de traduction
Université de Montréal
C.P. 6128, Succursale Centre-Ville
Montréal (Québec) H3C 3J7 — CANADA
Email: polguera@ere.umontreal.ca

1 Aim of this paper

This paper is an introduction to computer Text Generation, a discipline that has become one major area of Research and Development in natural language processing. I have tried to make this text as compact and simple as possible, in order to provide the reader with some sort of "popularization" of the concepts and techniques involved in Text Generation. In order to do so, I had to ignore a number of interesting and important considerations and I beg the more experienced readers to forgive me for the somewhat superficial nature of the present article. In fact, this article will probably be of no use to those already familiar with Text Generation: I have written it more for the benefit of non-specialists interested in knowing how Text Generation can be applied to the solving of practical problems, in the context of linguistic engineering. Among the many things one will not find in this paper, let me mention the following:

- I will not enter into much theoretical consideration, focusing instead on practical aspects of Text Generation. Nevertheless, some pointers will be given to theoretical problems and possible solutions.
- For lack of space, I will not include here a history of the research in Text Generation.
- I will not try to be completely objective, as pure objectivity (whether it can exist or not) would not necessarily help the reader forge a valid opinion about the importance of Text Generation for linguistic engineering. Whenever "opinions" will be expressed, I will make my own biases as explicit as possible in order to allow the reader to happily agree or disagree with what I say.

The remainder of this paper is divided into three sections. Section **2** offers a very general definition of what is meant here by *Text Generation*. Section **3** is a brief (and hence far from complete) presentation of the state of the art in Research and Development in Text Generation. Finally, Section **4** will try to make some predictions of what the future of Text Generation will be. In order to do so, I will start this final section with an estimation of the main limitations of current Text Generation systems.

2 WHAT IS TEXT GENERATION?

The infiltration of computer science in all fields of human activity had a significant side effect on the language of sciences and technologies: it created a terminological mess. New fields of activity emerged, and new terms were coined to designate these fields and the concepts they carry. Things went so fast, with such significant absence of coordination between people involved, that "new" sciences and technologies — i.e. those characterized by the intensive use they make of computers — found themselves caught in an intricate web of overlapping, contradicting and sometimes redundant terminology. Much toner powder (one no longer says *ink*) is spent on arguing about terminological problems, and, more importantly, people can happen to find or lose their jobs thanks to the ambiguity that reigns in the naming of new technologies.
Text generation is one of those many new terms, and it corresponds to one of those many new fields of Research and Development (R&D). As it identifies the main topic of the present paper, I find myself compelled to start with a bit of terminological definition (**2.1**). I will then proceed with a more technical look at what type of processing and knowledge Text Generation may entail (**2.2**). Finally, I will indicate what makes Text Generation a very specific domain of research and technological development (**2.3**).

2.1 General definitions

Before being a domain of R&D, Text Generation is, first of all, a type of human intellectual processing: when writing the present paper, I perform text generation. This activity can loosely be defined as the production of texts — i.e. coherent sets of communicatively connected sentences — from "conceptual structures." (In order not to delve here into philosophical considerations, this term will be taken for granted, with no further explanation.) Of course, the minimal set of sentences generated can contain only one element. For instance, when a friend bumps into

you on the street and says *Hi!*, he does perform text generation. Although not connected to any **linguistic** context, this simple text is nonetheless communicatively coherent as it connects adequately to the pragmatic situation in which it is uttered.

So much for "natural" Text Generation. We can now slightly modify the above definition in order to define "artificial" (or automatic) Text Generation, which is what concerns us here:

> Text Generation (hereafter, TG) is the automatic production, by means of computer programs, of texts — i.e. coherent sets of communicatively connected sentences.

From this, we can infer that TG is a sub-field of Natural Language Processing — computer processing of natural language. As a research field, TG can be involved in such domains as computational linguistics, computational psychology (a.k.a. cognitive sciences) and so forth. On the development side, TG is used in language engineering of all sorts, and it is on this that I will mainly focus in this paper.

To conclude, I should mention Artificial Intelligence (AI), which is somewhere along the spectrum between research and development. Some people see it as an "application" field and others as a research and theoretical one. This debate should not concern us here, and let us use AI to mean 'automatic modeling of intelligence in a research or development context;' a very broad field of investigation, indeed.

2.2 Types of processing and knowledge involved

The "machine" that performs TG — a text generator — takes as input sets of data and gives as output texts in natural language. This is the blackbox view of TG, and it is actually technically possible for a program to perform TG without, or with very little, internal modeling of the TG process itself. It is typically what happens in the development of prototype systems serving specific purposes: feasibility studies, additional utility modules that are part of more sophisticated programs, etc. However, "real" TG is always based on a more or less stratificational structuring of the TG process.

There is a general consensus on the fact that TG involves at least the two following processes:

1. **Planning**, which is the computation of the content and structuring of the text to be produced from input data;

2 **Realization**, which is the computation of the actual linguistic forms from a representation of their semantic content.

The corresponding components of TG systems are labeled, respectively, **planner** and **realizer**. This terminology is one among many; for example, **strategic component** (for planner) and **tactical component** (for realizer) is an alternative terminology made popular by K. McKeown — see [McKeown 1985].

I will propose a fictitious example of text generation in order for the reader to have a better understanding of what is involved in the planning and realization processes. Let us imagine a hypothetical text generator, HypoTG, that would be connected to a Unix operating system and would allow users to get a concise report on who is doing what on the system, at the present moment. For those familiar with Unix, such a report would be the natural language equivalent of the output of the `w` Unix command, which is illustrated below:

```
% w
  4:02pm  up 10 days, 16:28,  16 users,  load average: 2.09, 2.03,
2.28
User     tty from            login@   idle   JCPU   PCPU  what
brindav  q9  moliere.BB.URE   3:55pm             3      3  pine
cortomal q5  racine.BB.URED  3:53pm             46     13  kermit
blanseca q23 lambda.BB.URED 12:30pm     34   1:28     34
/usr/local/DB
larsong  q7  ffn0-0230v.rlm  3:51pm      1     12         -csh
palmerj  q3  moliere.BB.URE  3:55pm                     pine
supermar q8  131.123.131.96  3:35pm             36      5  tar xf ad
bergmang q10 omnibus.BB.URE  3:21pm      3      26     13  lynx
cornelij q1  lambda.BB.URED 12:30pm     19  32:10      7  -csh
grubermj q2  racine.BB.URED  2:01pm             55     54  emacs aba
bananar  q15 tghb1s.DE.URED  2:49pm             26     26  pine
robin    q17 racine.BB.URED  2:49pm             35     35  slirp
warshaw  q19 milou.BB.UREDi  3:36pm             14      1  elm
polguera q11 vegepate.BB.UR  4:01pm              4         w
elses    q12 milou.BB.UREDi  3:04pm             31     31  slirp
baincolo q29 top51.CPASMUR.  2:16pm   1:21      11      2  -csh
timl     q16 vegepate.BB.UR  3:29pm              6      6  pine
%
```

Figure 1: Sample output of the `w` Unix command

The data given in Figure 1 will be the input to HypoTG. The first things this system will do is to analyze this data, extract what it believes is particularly relevant, discard what is not essential, infer pertinent information, structure the information into a coherent set of messages, etc. In other words, HypoTG will **plan** the content of the report. This content could take the form of a series of formal expressions, each of these expressions encoding a representation of the meaning of a linguistic message HypoTG will have to realize:

```
(mes1  (connect  user:x time:now)
       (number   x 16)
       (AND    (active   user:all  time:now)
               (except   user:[larsong, bergmang, cornelij,
                                        blanseca, baincolo])))
(mes2  (be-idle  user:last-ref duration:(> 1:00)  time:past))
(mes3  (connected  user:x time:now)
       (name   x  "Tim Lawrence")
       (AND    (use    user:x-ref  program:pine  time:now)))
etc.
```

Figure 2: Sample output of the planning phase of processing

As shown in the above figure, HypoTG has to possess some knowledge about each user, and about what can make a given message particularly interesting. For instance, HypoTG may know that one particular user — Tim Lawrence (i.e. `timl`) — is regularly exchanging emails with the user who executed the `w` command and that it may be relevant to inform this latter that Lawrence is presently connected.

HypoTG will take each individual message in Figure 2 separately and translate it into a grammatical English sentence, thus **realizing** the text. In order to do so, it will use grammatical and lexical knowledge, and procedures to activate such knowledge for the synthesis of linguistic expressions. The resulting report could look as follows:

```
Sixteen users are connected – all are active except larsong,
bergmang, cornelij, blanseca and baincolo.  The latter has been
idle for more than an hour.  Tim Lawrence is presently connected,
using pine. (...)
```

Figure 3: Sample report

The above example, though fictitious, is perfectly realistic and gives a rough idea of what type of processing is involved in both planning and realization. It also demonstrates how useful TG can be from a practical point of view: there are many contexts where concise and informative natural language texts are preferred to “flat” collections of raw data — more on this in **3.2** below. This naive example is of course not sufficient to show how complex actual TG is to implement. There exist many detailed descriptions of TG systems. Readers may consult [Dale 1990] for a brief but thorough and clear description of a TG system. The system R. Dale describes is called EPICURE; it generates English recipes from:

- knowledge bases of ingredients and basic “cooking actions;”
- a model of the (cooking abilities of the) reader of the recipe.

EPICURE — developed with the aim of studying the generation of referring expressions — is a good illustration of TG applied to the production of instructional texts, which is one of the many possible applications of TG (see **3.2** below).

A number of projects — see, for instance, [Feiner and McKeown 1990] — are exploring the integration of TG in more global environments, where non-linguistic modes of communication (graphics, sounds, etc.) are involved. We witness increasing interest in this issue as modern computers are developed with operating systems that are more and more based on graphical and even audio interfaces. The report of Figure 3 could for instance be expressed vocally, provided that our HypoTG incorporates a voice synthesis component. Additionally, it could be coupled with the production of graphics, supplying the user with additional information, to which the generated text could make reference.

Although I have introduced the planning and realization processes as sequential operations, they can actually be performed simultaneously. Researchers do not agree on whether a "pipeline" (i.e. sequential) architecture is psychologically valid. But most developers of TG systems agree on the fact that such architecture is computationally the most tractable one. An opinion shared by many is that "natural" TG is probably more sequential than text understanding (where the interaction between the use of linguistic and non-linguistic knowledge is obviously non-sequential). It is therefore rather satisfactory to treat planning and realization in a sequential way. Nevertheless, nothing proves that the optimum treatment should be 100% sequential. Most planning tasks are in fact contingent upon the lexical and grammatical characteristics of the targeted language, and the need for more planning may arise in the process of realizing a given linguistic structure. To take a concrete example, the computation of grammatical tense involves, beside grammatical knowledge, some (non-linguistic) knowledge about the situation one is referring to; but it is only once you know that a given atom of information will be expressed in English by a verb in a very specific syntactic position that the need for computing its grammatical tense arises. Traditionally, this problem is circumvented by computing the relevant information each time it **may** have to be used by the realizer — this is what HypoTG did in the above example. However, one can clearly see how artificial this approach is. It is not satisfactory from a theoretical point of view, but it is the one that proves the most efficient in actual TG applications, for now.

The realization and planning processes can each be broken up into subprocesses. In fact, they have to, if one wants to target systems that are not of the blackbox type. However, the stratification of planning and realization are different issues, as I will briefly try to demonstrate.

It is very difficult to have a general theory of the planning process as this process is very dependent on what type of data it takes as input and what type of linguistic semantic/conceptual representations it provides as output. The former is contingent upon the domain of application proper, and the latter is contingent upon the theoretical choices made by developers for the structuring of the more linguistic process of realization. There are thus too many unknowns and this forces developers and researchers to considerably adapt their strategies for each different application they work on. What is usually theorized are general conceptual ontology or formal modes of knowledge representation.
In contrast, the realization process can be more easily modeled based on a given linguistic approach. In this case, the linguistic theory that is used as background for the development of the realizer will impose a given structuring. There will be more on this below.

2.3 Specificity of Text Generation

TG is indeed a very specific type of natural language processing. Without entering into a lengthy study of this specificity, I will try to illustrate it by comparing TG with its two “siblings” in the Natural Language Processing family: Text Understanding (**2.3.1**) and Machine Translation (**2.3.2**).

2.3.1 Text Generation vs. Text Understanding

Text Understanding is more or less the converse process of TG; namely: the execution of a task (any task) by a computer as a result of the interpretation of a text in natural language. Very often, the “text” involved in Text Understanding will be a single sentence: a natural language query to a database, an instruction to a mechanical or logical device (cf. spoken instructions to a computer operating system), etc. Historically, much more emphasis has been put on Text Understanding than on TG. There are probably many reasons for this, but three are worth mentioning here:

1. Most formal linguistic theories since the fifties have focused on syntactic rather than semantic phenomena, putting the emphasis on an analytical rather than synthetic view of linguistic structures (more on this in **4.1.2.1** below).

2 The first natural language processing application to be explored with ambitious projects was Machine Translation, and the initial stage of the translation process is of course the interpretation of the source text. It is also this stage that posed most problems to researchers and developers, thus the emphasis on parsing technologies, that are at the center of the problem of Text Understanding.

3 A more philosophical, but not necessarily less relevant explanation, is that it is a very common human attitude to enjoy talking and being understood, much more thrilling for many than to listen and understand. In the early stages of Artificial Intelligence, people were dreaming of a slave machine that could passively listen and understand them, this being the utmost degree of intelligent behavior that could be expected from a machine. It was only later that people understood that an intelligent machine should also be a machine that has something to say.

Interestingly, while Text Understanding has been for many years the main focus of research, it is probably at the same time the most difficult problem to address well. It is in some sense easier to control the behavior of a TG system as the problem of unknown words or unknown grammatical constructions is virtually non-existent in TG.

2.3.2 Text Generation vs. Machine Translation

As I have mentioned, Machine Translation entails (at least) a two-fold process: it involves (i) the analysis of a source (input) text and its translation into some form of intermediate representation, and (ii) the synthesis of the target (output) text from this intermediate representation. The following figure presents a condensed view of how TG, Text Understanding and Machine Translation relate, through the various processes they model:

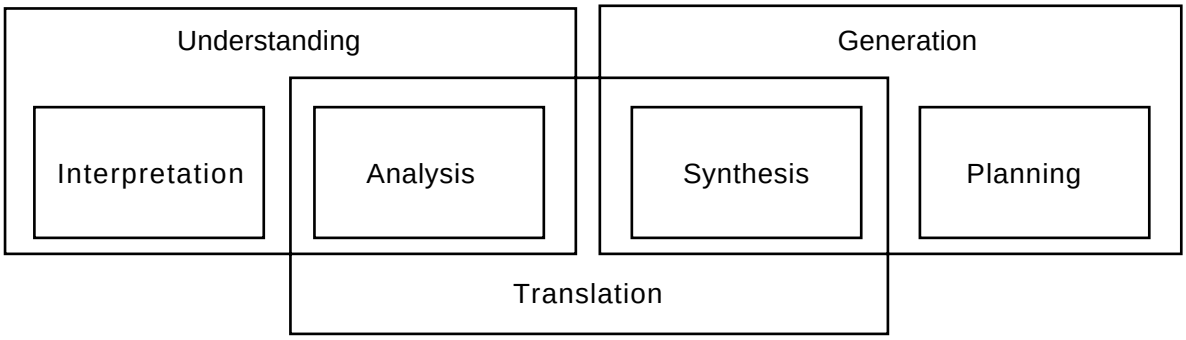

Figure 4: Relationship between various aspects of natural language processing

The above figure is of course an outrageous simplification. To take only one example, it is not clear at all that translation is made up of only analysis and synthesis. In actual fact, **natural** translation necessarily contains an interpretation process, which contradicts Figure 4. But in spite of its simplistic nature, this figure can help in understanding the relative specificity of TG. It can be noted that, from a research point of view, TG is particularly relevant for anyone who is interested in semantic phenomena. The bulk of the early work in Text Understanding has been done trying to avoid "messing up" with the semantics of natural languages. This just cannot happen in TG, which is centered around the production and expression of linguistic meanings.

To conclude this section, it is worth noticing that Machine Translation systems can actually make use of (parts of) TG systems. For instance, the PANGLOSS Spanish-to-English Machine Translation system — whose description can be found in [Wilks *et al.* 1996:229-234] — makes use of the PENMAN English text generator (see **3.1** below) to perform the synthesis of English sentences. Many connections, from an application point of view, can thus be found between Machine Translation and TG. It has even been demonstrated in recent works (see **3.2** below) that **multilingual** TG can sometimes be a very effective alternative to Machine Translation.

3 From research in Natural Language Processing to Language Engineering: the R&D situation

I will not try to present here a history of the work done in TG. Such a review of TG projects would take too much space, and is not in the scope of the present paper. Furthermore, the work has already been done on several occasions — for instance, a review that takes the viewpoint of systemic-functional linguistics can be found in Chapter 3 of [Matthiessen and Bateman 1991]. The goal of this section is mainly to present the connections that exist between research work in TG and actual or potential applications that can be derived from it.

3.1 A brief overview of past research in Text Generation: two basic references

Let us look at two "big names" in TG, two projects that have made history. I have made my selection of these projects based on the following criteria: research projects (i) that are known by virtually everybody active in the field of TG, and (ii) whose design and performance are still of interest now — in spite of the progress achieved in the field of TG since these systems were programmed.

K. McKeown's **TEXT** generator

I have already mentioned (Section **2.2**) K. McKeown's Ph.D. research, published in [McKeown 1985]. She programmed the TEXT TG system to explore strategies for planning communicatively coherent texts. In that sense, the main focus and interest of her research lies at the planning level, where she proposed the use of so-called **rhetorical schemata** to build and structure the content of texts. TEXT was one of the first serious attempts to give a theoretical basis to the communicative organization of texts in TG. Since this pioneering research, much work has been done on the planning of well-structured texts. To make it short, I will simply mention here the work done using [Mann and Thompson 1988]'s **Rhetorical Structure Theory**: see, for instance, [Hovy 1988].

The systemic PENMAN generator

A general presentation of the PENMAN system can be found in [Matthiessen and Bateman 1991]. In my opinion, PENMAN is significant in that it probably represents the first attempt at building a general-purpose TG system whose realization component would embody a fairly complete grammatical description of a natural language. NIGEL, the English grammar embedded in PENMAN, is not only huge (by TG standards), but it is also encoded in a fairly explicit way, using theoretical principles stated in systemic-functional linguistics — see [Halliday 1985]. It is remarkable that PENMAN has actually provided systemic linguists with a tool for exploring and testing their theoretical hypotheses. In actual fact, the main aim of the developers of PENMAN was precisely to explore the theoretical framework of systemic-functional linguistics. Much work has been done in connection with the approach taken by PENMAN, with many projects aiming at generating texts in languages other than English — see, for instance, [Bateman 1986, 1991] for Japanese, and [Steiner *et al.* 1990] for German. It is worth mentioning that the PENMAN Lisp code has been made available for research purposes by the Information Sciences Institute (ISI) of the University of Southern California.
In spite of its many qualities, two main critiques can be made to PENMAN:

1 The system, as it is, does not come equipped with sophisticated planning procedures that would allow for the building of communicatively coherent texts. In that sense, PENMAN can be directly used as a realizer, but additional components have to be added in order to obtain an actual application that would perform the complete TG task.

2 The lexicon that is included in NIGEL is pretty limited (when compared to the sophistication of the grammar itself). No extensive lexicographic work has been done within the PENMAN project.

3.2 Technological transfer to Language Engineering

Most research in TG is done with a certain type of application in mind. Looking at what has been done in the past fifteen years or so is enough to demonstrate how broad the range of applications of TG can be. I will limit myself here to only three different types of applications, for which I will mention examples of actual operational systems, in addition to experimental prototypes.

Generation of reports

K. McKeown's TEXT is to be listed here. It is part of an information retrieval system which allows the user to access information contained in a portion of the US Office of Naval Research database, by producing short reports in answer to the user's queries. For instance, in answer to the query *What is a ship?*, TEXT will extract the propositional information (A) from the database and express it as (B) — see [McKeown 1985:52, Figure 2-14]:

(A)
```
(identification SHIP WATER-VEHICLE (restrictive TRAVEL-MODE
SURFACE)
    (non-restrictive TRAVEL-MEDIUM WATER))
      (evidence based-db SHIP (TRAVEL-MODE SURFACE) (HAVE DRAFT)
    (HAVE DISPLACEMENT))
```

(B)
```
The ship is a water-going vehicle that travels on the surface.  Its
surface going capabilities are provided by the DB attributes DRAFT
and DISPLACEMENT.
```

The generation of reports is typically the type of application that has been exemplified in Section **1**, with the fictitious HypoTG system. There has in fact been actual work done on the production of reports on the activity of computer operating systems: for instance, the GOSSIP system, whose Meaning-Text realizer is described in [Iordanskaja *et al.* 1991]. The work on GOSSIP led to the development of an operational TG system for the production of bilingual (English and French) weather forecasts: FoG, described in [Kittredge and Polguère 1991, Kittredge *et al.* 1994]. FoG is presently used by Environment Canada, as the TG component of the Forecast Production Assistant (FPA) — see [Goldberg *et al.* 1994].

FoG is a very typical example of how multilingual TG can be used as an alternative to Machine Translation. The texts that FoG directly produces in both English and French were previously manually written in English, then translated into French by the TAUM-Météo system. The operational success of FoG does not originate only from the fact that this system deals with a so-called sublanguage (weather forecast). It is also due to the fact that FoG was developed as an integral part of a more global "knowledge processing" environment: the FPA workstation. There will be more on this very important point in **4.1.1** below.

Generation of instructional texts and business letters

The automatic production of instructional texts has great potential for the development of TG applications. It has also been researched in the context of quite a few projects. I have already mentioned R. Dale's EPICURE system for the generation of cooking recipes (which is a particular case of instructional texts). Another example of research in this field is the PENMAN-based IMAGENE system, for the generation of technical manuals — see [Vander Linden and Martin 1995]. On the operational front, we find systems such as the AlethGen TG "toolbox" — see [Coch *et al.* 1995], which is used for the generation of business letters. (AlethGen's realizer is based on Meaning-Text linguistic principles.)

I put generation of business letters in the same category as generation of instructional texts because both applications share at least two essential characteristics:

1. They cannot be said to take as input simple formatted databases: text is here produced from representations of highly sophisticated types of knowledge.
2. To be operational, they almost necessarily need to incorporate some form of tailoring of the text, based on a model of the user (see **4.1.1** below).

Of course, there do exist obvious differences between the two types of TG, and I hope the reader will forgive me for the amalgam.

Computer-assisted language learning

There are many ways a TG system could be used in the context of Computer-Assisted Language Learning (CALL). For instance, students could interact with such systems in order to explore various ways of expressing their thoughts in a

given foreign language. TG strategies applied to CALL have already been explored and prototypes systems designed — see, for instance, [Zock 1992].
On the operational front … well, I don't know of any operational CALL system making use of TG techniques (any information on this is welcome!). But this is definitely a very important and interesting field of application that would deserve greater attention.

3.3 Non-linguistic realization

Being trained in linguistics, I tend to be more interested by TG systems in which actual linguistic knowledge is used in the realization process. But there are many instances of TG systems that do not implement this type of realization. These systems are of two types:

- experimental systems for which researchers are mainly interested in planning problems and do not want to spend much energy on the implementation of the realization process;
- operational systems where texts to be generated are so simple and linguistically poor that no sophisticated realizer is needed.

In both cases, the solution is to use fixed or semi-fixed sentence templates, the role of the realizer being then to "fill the empty slots." Such a technique, called ***template approach*** in [Reiter 1995], is of course rather uninteresting from a linguistic point of view. But from a practical point of view, it can prove very efficient, when linguistic competence is precisely not at stake. Developers of TG applications have to first study whether their applications will need a linguistic realizer or whether a template realization will do. This preliminary study should be based on, **at least**:

- the linguistic characteristics of the texts to be produced — whether stylistic, grammatical, lexical, etc. variations are needed or whether very repetitive types of texts are acceptable;
- the operational requirements of the application — whether the TG system will need maintenance or not.

An experimental assessment of three types of text production — non-linguistic ("fill-in-the-blanks"), mixed linguistic and template, and entirely manual — can be found in [Coch 1996]. Focusing on the writing of business letters, this study gives insights on (i) when (machine) TG can prove more effective than manual

redaction, and (ii) when sophisticated linguistic knowledge can be needed in engineered text generators.

4 NEW ORIENTATIONS IN TEXT GENERATION

Just looking at the proliferation of projects involving TG, one can be confident that it represents a domain of application with great potential. Nevertheless, TG, like most other branches of Natural Language Processing, seems to be progressing at a slower pace than one would have expected. In this final section, I will try to give some explanation for this and make some prediction on what the future of TG may be, in the global context of linguistic engineering.

4.1 What can explain the limitations of the present systems?

4.1.1 The Artificial Intelligence bottleneck

In **2.3.1**, I tried to answer the question of why TG has been slower than Text Understanding in taking off. Faced with the same question, R. Dale, C. Mellish and M. Zock proposed another, also very plausible, explanation:

> One possible reason for the imbalance is that there are few applications which are rich enough in terms of what they want to express to justify the construction of a facility which makes sophisticated use of natural language. This situation is changing; in particular, the increasing sophistication of intelligent knowledge based systems requires just such facilities.
>
> [Dale *et al.* 1990:2]

I cannot agree more with this explanation as it fits well with my own personal experience in the construction of TG applications. Having participated in the development of both operational and prototype systems, I notice that what really makes the difference is often whether a given TG system has its place in a more global knowledge-based system. In other words, very few TG applications are self-sufficient; in order to become operational systems, they have to fit into more global systems, that often require a more considerable amount of time and effort to develop than the TG component itself.

As long as machines have nothing or not much to say, TG will have great trouble finding its application in operational systems. Fortunately, as mentioned above by R. Dale *et al.*, the situation is evolving fast. What is needed now is a better coordination between researchers working on general purpose knowledge-based systems and those developing TG systems. Only a high degree of sophistication and coordination in the work on these two types of knowledge engineering can ensure the development of good operational TG systems.

Another limitation of most TG systems, that fits in this category of general AI problems, is the absence of high-level user modeling. It is a well-known fact, and all of us experience this everyday, that speakers tailor both the form and content of their speech depending on who they are talking to; this is an essential aspect of natural language communication. Without this tailoring of our speech, we would not be able to communicate properly. In the same vein, TG systems need to possess this ability to tailor their linguistic production in order to be efficient and usable in a broad range of contexts. This can only be achieved with a certain level of what is known in the AI framework as **user modeling**. Significant research work has been done on this topic — see, for instance, C. Paris's TAILOR system [Paris 1993] and the hearer model of R. Dale's EPICURE system [Dale 1990]. The benefits of such research are just beginning to surface in the context of actual TG applications.

4.1.2 The linguistic bottleneck

The content of this subsection is totally biased and subjective; but whether or not it is **the** truth, I believe that there is **some** truth in it. At least, the reader may be certain that everything that comes now (i) is not just my own personal opinion (it is shared by many others), and (ii) results from practical experience with both linguistic models and TG systems.

4.1.2.1 The non-generative generative grammars

Most so-called mainstream approaches to linguistic formalization, whether or not they claim to fit in the generative framework initiated by N. Chomsky, are in actual fact oriented toward the **analysis** (rather than synthesis) of linguistic utterances. This is a rather theoretical issue, which cannot be seriously discussed here, but to give a quick demonstration of what I have just stated, I will identify two types of linguistic models:

1. models that are designed in order to describe some linguistic constructions as grammatical and others as ungrammatical, these constructions being taken as points of departure for the description;
2. models that are designed to describe how given meanings can be expressed in natural languages.

The models of the first type are by nature analytical, while models of the second type are by nature synthesis-oriented. In my opinion, the focus on analysis, which is a characteristic of so-called generative approaches to linguistic

formalization, should seriously be questioned by both linguists and users of linguistic theories. As a matter of fact, a non-trivial number of projects in TG make use of non-generative approaches — e.g. M. Halliday's systemic-functional linguistics ([Halliday 1985]) and I. Mel'čuk's Meaning-Text Theory ([Mel'čuk 1981]) — that favor a synthesis-oriented view of language. This is not my own personal interpretation of the situation. For instance, C. Matthiessen and J. Bateman offer reasons for choosing the systemic paradigm in designing a TG system which are very similar to my own reasons for choosing to work with another non-generative framework, Meaning-Text theory:

> (...) systemic linguistics interprets and represents language not as a rule-system for generating structures but as a resource for expressing and making meanings. The interpretation of language as a *resource* makes a fundamental difference when we build a text generation system since the purpose of such a system is precisely to express and make meanings.
>
> [Matthiessen and Bateman 1991:4]

What has just been said here does not of course mean that only systemic-functional linguistics and Meaning-Text linguistics can be used to design TG systems. My aim is simply to emphasize the fact that not all linguistic approaches are equivalent and that the choice of which linguistic framework to adopt is one that has to be made after a great deal of thinking. Choosing a theoretical linguistic framework to develop a natural language processing system is a bit like choosing a tool to do handiwork: think first, or you may end up trying to drive in a nail with a toothpick. More importantly, no linguistic theory can pretend to cover equally well all aspects of linguistic knowledge and processing. Therefore, designers of natural language processing systems will often have to draw from different approaches, borrowing from each of them what they do best.

4.1.2.2 Missing words

One of the aspects of linguistic knowledge that most contemporary linguistic theories do not treat with sufficient care is the lexicon. For some obscure reason, the lexicon has been considered by many as some kind of appendix to language, with the result that formal linguistics mainly concentrated on sentence structure. As a result, we lack extensive formal descriptions of the lexicons of natural languages. All we have are “traditional” dictionaries. But while good dictionaries are very useful pieces of work, they do not provide, as they are, descriptions that are directly usable in the context of natural language processing. If we consider the following two facts:

1 language is before all a tool for expressing meanings,

2 the bulk of meanings expressed in languages are lexical meanings (as opposed to grammatical meanings such as grammatical tenses, aspects, etc.),

it is obvious that the lack of extensive formal descriptions of lexicons is a major bottleneck in the work in TG, and natural language processing in general. Without solving this problem, one is bound to work only with very limited applications. On experiments in TG which involve the use of more sophisticated lexicons, see [Wanner 1994, Wanner and Bateman 1990] and [Iordanskaja *et al.* 1996].

4.2 New developments: Through the lexicon bottleneck

There are two main solutions to remedy the lexicon bottleneck problem:

1 on the linguistic side, put very strong emphasis on the writing of extensive formal dictionaries that can serve as bases for the lexicons of natural language processing systems;
2 on the AI side, try to find ways to automatically process traditional dictionaries in order to (i) extract the information needed by natural language processing systems, and (ii) formalize this information in a computer-tractable format.

Many insights on this can be found in [Wilks *et al.* 1996], where various approaches to the solving of the lexicon bottleneck in natural language processing are presented and analyzed.

In spite of the fact that TG has been gaining more and more importance in natural language processing, it is clear that its interest is still underestimated by researchers and developers. In major conferences on natural language processing — see for instance the *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)* — the bulk of the papers are still focusing on an analysis rather than synthesis orientation. By *analysis orientation*, I mean such topics as speech recognition, morphological tagging, parsing, etc. Nevertheless, I will venture to make the following predictions:

- The work on building extensive linguistic descriptions will intensify, as more and more people become aware that this is a prerequisite to good TG and natural language processing.
- In this context, the task of building huge lexicons that are fully integrated in, and compatible with, grammatical descriptions, will sooner or later provide

developers of TG systems with ready-to-use linguistic tools that will drastically increase the potential of TG in terms of actual applications.

In that sense, the future of TG cannot be dissociated from the future of natural language processing in general.

ACKNOWLEDGMENTS

Many thanks to José Coch and Helen Lim for their insightful comments on a draft version of this paper.

REFERENCES

Bateman, J. A. 1986. Text planning for a systemic-functional grammar of Japanese. Technical report, Department of Electrical Engineering, Kyoto University, Kyoto, Japan.

Bateman, J. A. 1991. Uncovering textual meanings: a case study involving systemic-functional resources for the generation of Japanese texts. In C. L. Paris, W. R. Swartout and W. C. Mann (eds.): *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Boston *et al.*: Kluwer, 125-153.

Coch, J., R. David and J. Magnoler. 1995. Quality test for a mail generation system. *Proceedings of Linguistic Engineering'95*, Montpellier, France.

Coch, J. 1996. Evaluating and comparing three text-production techniques. *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, Copenhagen, Denmark, 249-254.

Dale, R. 1990. Generating Recipes: An Overview of Epicure. In R. Dale, C. S. Mellish and M. Zock (eds.): *Current Research in Natural Language Generation*. London *et al.*: Academic Press, 229-255.

Dale, R., C. S. Mellish and M. Zock (eds.). 1990. *Current Research in Natural Language Generation*. London *et al.*: Academic Press.

Feiner, S. and K. R. McKeown. 1990. Coordinating Text and Graphics in Explanation Generation. *Proceedings of the 8th National Conference on Artificial Intelligence*, Boston, Mass., 442-449.

Goldberg, E., N. Driedger and R. I. Kittredge. 1994. Using Natural-Language Processing to Produce Weather Forecasts. *IEEE Expert*, April 1994, 45-53.

Halliday, M. A. K. 1985. *An Introduction to Functional Grammar*. London: Edward Arnold.

Hovy, E. H. 1988. Planning Coherent Multisentential Text. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, 163-169.

Iordanskaja, L., R. I. Kittredge and A. Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In C. L. Paris, W. R. Swartout and W. C. Mann (eds.): *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Boston *et al.*: Kluwer, 293-312.

Iordanskaja, L., M. Kim and A. Polguère. 1996. Some Procedural Problems in the Implementation of Lexical Functions for Text Generation. In L. Wanner (ed.): *Lexical Functions in Lexicography and Natural Language Processing*. Amsterdam/Philadelphia: John Benjamins, 279-297.

Kittredge, R. I. and A. Polguère. 1991. Dependency Grammars for Bilingual Text Generation: Inside FoG's Stratificational Models. *Proceedings of the International Conference on Current Issues in Computational Linguistics*, Penang, Malaysia, 318-330.

Kittredge, R. I., E. Goldberg, M. Kim and A. Polguère. 1994. Sublanguage Engineering in the FoG System (Poster paper). *Proceedings of the 4th Conference on Applied Natural Language Processing*, Stuttgart, 215-216.

Mann, W. C. and S. A. Thompson. 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *TEXT*, 8:3, 243-281.

Matthiessen, C. M. I. M. and J. A. Bateman. 1991. *Text Generation and Systemic Functional Linguistics: Experiences from English and Japanese*. London/New York: Pinter.

McKeown, K. R. 1985. *Text generation: Using discourse strategies and focus constraints to generate natural language text*. Cambridge: Cambridge University Press.

Mel'čuk, I. A. 1981. Meaning-Text Models. *Annual Review of Anthropology*, 10, 27-62.

Paris, C. L. 1993. *User Modelling in Text Generation*. London/New York: Pinter.

Reiter, E. 1995. NLG vs. Templates. *Proceedings of the 1995 European Natural Language Generation Workshop*.

Steiner, E. H., J. A. Bateman, E. Maier, E. Teich and L. Wanner. 1990. Of mountains to climb and ships to sink: generating German within a functional approach to text generation. Technical report, GMD/Institut für Integrierte Informations- und Publikationssysteme, Darmstadt, Germany.

Vander Linden, K. and J. H. Martin. 1995. Expressing Rhetorical Relations in Instructional Text: A Case Study of the Purpose Relation. *Computational Linguistics*, 21(1), 29-57.

Wanner, L. 1994. On Lexically Biased Discourse Organization in Text Generation. *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, 369-375.

Wanner, L. and J. A. Bateman. 1990. A Collocational Based Approach to Salience-sensitive Lexical Selection. In *Proceedings of the 5th International Workshop on Natural Language Generation*, Dawson, 31-38.

Wilks, Y. A., B. M. Slator and L. M. Guthrie. 1996. *Electric Words: Dictionaries, Computers, and Meanings*. Cambridge, Mass./London, England: The MIT Press.

Zock, M. 1992. SWIM or sink: the problem of communicating thought. In Swartz, Merryanna L. and Masoud Yazdani: *Intelligent tutoring systems for foreign language learning*, Berlin: Springer-Verlag, 235-247.