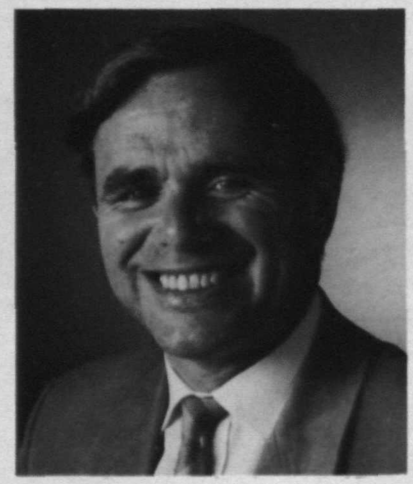
Beyond Word Processing


Gerard Kempen

Gerard Kempen is professor of psycholinguistics at the University of Nijmegen in The Netherlands. He is working on theoretical and applied aspects of natural language processing, in particular on the development of an intelligent author system.

Word processing is now a major end use application of computers. 'Author systems' which can treat text as a linguistic structure as well as a string of characters to be manipulated will make today's word processing systems seem primitive. While such systems remain on the horizon, intelligent programming tools and translation aids are already here.

The 1980s have seen the advent of the first commercial applications of artificial intelligence research. The two prime examples are expert systems and natural language processors. These developments have already begun to affect the design of office computers. Future office workstations will have some 'understanding' of office procedures and of the goals of office workers and their organisation. Also, they will have natural language interfaces. This will enable office workers to interact with the workstation in their native language rather than in some artificial language. Moreover, the workstation will act as a linguistic consultant in facilitating the preparation, manipulation and translation of full text documents.

The present situation

While the full realisation of such capabilities in marketable office computers is unlikely before the 1990s substantial portions are already within reach of present day microcomputers, notably English language dialogue systems, semi-automatic translation systems, and linguistic editors. By the latter I mean special software which runs on word processors and assists writers and typists in checking for proper spelling, grammar and style. Good examples are the Writer's Workbench which was developed at Bell Labs, and IBM's CRITIQUE. Various microcomputer based semi-automatic translation systems are currently on the market. They offer word processing facilities specifically tuned to the translator's job and produce the first draft of a translation which is then edited by the translator. Some of these microcomputers can serve as front end processors for more advanced translation systems based on mini or mainframe computers.

Machine translation

Reports on the cost effectiveness of machine translation have already appeared. In 1983 the first conference on applied natural language processing was organised by the American Organisation for Computational Linguistics (ACL). There Jonathan Slocum presented the results of a comparative study which pitted the normal procedure (human translation and post-editing) against a machine translation procedure. (The machine output was revised by a human translator.) The total cost of semi-automatic translation, including software and hardware cost, turned out lower than normal translation. The translations produced under the two conditions were judged to be of comparable quality. An important practical advantage of machine aided translation will be a reduction in the time required for translation projects.

Dialogue systems

A useful scheme for evaluating the capacities of natural language dialogue systems has been prepared by a team of experts under the auspices of ACL. They distinguish three levels of linguistic skill at which dialogue systems may operate. At the lowest level (level one) we find all the systems currently available for practical use, such as INTELLECT and the French language dialogue system SAPHIR. Level two systems so far exist solely as laboratory prototypes, and work on level three systems has only just begun.

The following summary of

the three levels may serve as a point of departure for further discussion. Level one systems contain a basic package of software modules for:

- the analysis and interpretation of recurring words, word groups and sentences, especially interrogatives
- generating sentential answers
- extending and adapting the vocabulary of words and idioms the system can draw upon
- constructing queries in the formal language used by the database management system.

Level two systems offer powerful methods of solving referential descriptions (ie various types of noun phrase, referring to objects, states and events in the domain with which the database is concerned). They incorporate the following additional design principles:

- the linguistic modules can to a high degree be guided by 'world knowledge', ie a representation of the database's domain of interest
- the discourse context (ie preceding questions, commands, answers, etc) is retained and exploited to the full
- input sentences are not translated straight into the formal database query language but into a logical interlingua, often a form of first order predicate calculus. All sorts of linguistic phenomena can now be dealt with in a more general and satisfactory manner. Moreover, the step of translating from the logical interlingua to the formal query language is relatively simple.

Facilities typically offered by level one dialogue systems*

1 Answering factual questions relating to data in common types of database system.
2 Co-ordinating data files (eg 'What is Smith's location?' is expanded into 'What is the location of the department of Smith?').
3 Resolving simple cases of pronominalisation (ie finding out what pronouns refer back to; difficult cases are still beyond most systems' reach).
4 Handling simple cases of ellipsis. (Elliptical questions are incomplete and refer back to earlier sentences in the dialogue: 'Where is John? ... and Peter?.)
5 Giving co-operative answers to 'null questions' (eg to the question 'How many copies of book X have you in stock?', the system might answer not with 'None' but with 'That book is not known to me' or 'That book is sold out').
6 Enriching linguistic knowledge through interaction with the user (eg 'Define "JD" as "Jefferson Davis Jones". Let "Q1 Smith salary" = "What is the salary of employee Smith?". Q1 JD age?'.)
7 Paraphrasing input sentences (queries, commands) so that the user has some check on whether they have been correctly interpreted.
8 Correcting spelling errors in input sentences, and reacting meaningfully to ungrammatical input.
9 Updating data using commands in natural language (eg 'Change Bob Day's location to Building 7').
10 Answering 'metaquestions' such as 'What are the permissible values for employee job titles?'; 'How up-to-date are the sales data?'; What information is in the database?'; 'Can you handle relative clauses?'

* (Adapted from Hendrix, 1982)

Level two

These further basic principles enable dialogue systems to interpret referential descriptions. Dialogue systems at level two are considerably better equipped to determine the correct reference of noun phrases than their predecessors. Suppose a car repair information system is asked: 'If I want to top up the water, where do I find the filler cap?' The decision that it is the cap on the radiator and not the petrol tank is based not on linguistic rules but on knowledge of cars.

Nor can the discourse context be disregarded when it comes to resolving referential descriptions. Take the stock control database of a bookshop when it is commanded to 'Give titles in stock and prices'. If the dialogue system is functioning at level one it will produce a complete list of all the books in stock. A level two system, by contrast, takes account of what has gone before. For example, after having been asked 'Do you know any of Virginia Woolf's books?' the response to the 'Give titles . . .' command will confine the list to works by that author. And the noun phrase 'the same' in a subsequent command, 'Now the same for Iris Murdoch' will be interpreted correctly as well. (This is also an example of pronominalisation, which would stump a level one system: 'the same' refers not to an object mentioned earlier but to an earlier sentence in the dialogue.)

Level three

Dialogue systems of level three will be capable of reasoning about mental states, such as goals, plans and beliefs. They will have to draw on an important extra source of information: knowledge of goal directed behaviour. Starting from needs and goals that they attribute to users, to themselves and possibly to third parties, they will be able to evolve plans and proposals which will as far as possible accord with the interests of those parties. The answers of such dialogue systems are based on such plans. Robert Wilensky of the University of California at Berkeley is one of the first to have started on the construction of such a system. He is developing a program which can act as a 'consultant' to users of the UNIX operating system, the chief purpose of the consultant being to help inexperienced users on their way.

We have now looked at a rising curve of sophistication in dialogue systems available now or in the near future. This does not mean that in a few years we shall be able to communicate with computers just as we do with people. Far from it. The concepts underlying the design of dialogue systems have improved considerably in recent years, and this vertical trend of deepening knowledge will continue in the future. But this does not automatically lead to a broadening of knowledge and skills within particular levels. Modern dialogue systems always operate on a very narrow knowledge base, and extension sideways is no trivial matter. It is labour intensive because large volumes of knowledge have to be inventoried, analysed and encoded without error. Moreover, it makes heavy demands on memory, which can lead to forbiddingly long response times.

It should be kept in mind that these limitations apply with equal force to other applications of natural language technology. For example, fully automatic translation is impossible without a deep conceptual understanding of the source text. This, in turn, requires a great deal of domain knowledge.

Examples of a dialogue with the Unix Consultant developed by Wilensky

User:	How do I delete a file?
UC:	Typing 'rm filename' will remove the file with name filename from your current directory.
User:	A directory?
UC:	Typing 'rm -r directoryname' will recursively delete the directory with name directoryname and all the files and directories contained in the directory.
User:	How should one edit a file?
UC:	Typing 'ex filename' will get you into the ex editor; typing 'emacs filename' will get you into the emacs editor.
User:	What is the best way to create a file?
UC:	The best way to create a file is to use the editor, ie 'ex filename'.
User:	A directory?
UC:	Typing 'mkdir directoryname' will create a directory with name directoryname in your current directory.
User:	How can I find out how full the disk is?
UC:	Typing 'df' will tell you how full the disk is.
User:	What is a search path?
UC:	A search path in Unix is a list of directories in which the operating system searches for programs to execute.

Author systems

I propose 'author systems' as a collective term covering all sorts of software tools which help authors in writing, editing and manipulating texts composed of natural language sentences. A major drawback of present day word processors and text editors is their almost complete ignorance of language structure. What linguistic knowledge they have is restricted to spelling and hyphenation. However, this situation is about to improve. The first indications are the various software packages capable of providing textual critiques, among them the Writer's Workbench. Another example is CRITIQUE, currently under development at IBM. Authors writing in English can utilise these tools to check their spelling, grammar and style. The facilities offered range from hypenation, punctuation and spelling control to the correction of points of grammar and style (split infinitives, verbosity, overly long sentences, abstract language, too many passives, missing subject-verb agreement, and other grammatical violations).

The end product of such programs is not necessarily accurate and foolproof since it is based on a rather incomplete linguistic analysis of the input text. More powerful techniques for syntactic parsing and conceptual understanding are clearly needed (a situation akin to that currently affecting dialogue systems). Despite such shortcomings, professional writers and office workers have been seen to respond positively to the new writing tools. This suggests that the limitations of present day language technology need not stand in the way of useful applications in the area of intelligent word processing.

Textual modification

An example of this arises with textual modification at word level. If you want to pluralise a noun which occurs frequently in a text, then current word processors allow you to change this noun to its plural form by issuing a single command. However, the linguistic changes to other parts of the sentence which this modification entails cannot be computed automatically. For example, 'this document' would have to change into 'these documents'. Additionally, in any sentence where the pluralised noun phrase plays the role of grammatical subject, finite verbs would need to be pluralised as well in order to maintain subject-verb agreement. Furthermore, all pronominal references ('it') to the plural noun phrase would have to be replaced (with 'they' or 'them', depending on syntactic function). In current word processors this is done by hand. In languages with a rich morphology like French and German, this is a time-consuming process and one prone to error. In an author system such as I envisage, such changes would occur automatically.

At sentence level, an author requires a simple means for altering word and phrase order in accordance with the rules of the language (as in the differing word orders in main and subordinate clauses in German and Dutch), for active/passive transformation, for adding co-ordinated constituents, and for suggesting paraphrases. Such facilities necessitate a sophisticated user interface which enables easy access to the linguistic structure of each sentence. For instance, writers should have the possibility of viewing 'syntactic trees' on a high resolution screen, and of addressing not only letters, words and lines as in current word processors, but also syntactic units such as noun phrases, clauses, and other parts of speech.

Possible applications

Author systems will be particularly useful at the discourse level. A relatively simple application is the automatic generation of large numbers of individualised documents such as business letters, which need more linguistic variation than can be handled by current menu-based report generators using templates. Consider the following template:

'Following your order *invoice number* which was placed on *date* we have sent you *number* copies of *item name*.'

Now suppose a customer has sent two invoices that came in on two consecutive days. The expanded form needed in this case could be a sentence like:

'Following your orders Z01245 and Z01246 placed on May 12 and 13 respectively, we have sent you 10 copies of manual A and one copy of manual B.'

In the author system I envisage, this amount of linguistic variation could be easily handled.

So far we have looked at the various natural language processing applications as separate software systems. Since there is a large amount of overlap between components, it seems worthwhile to try to integrate them. For instance, all applications need quick access to an online lexicon and a grammar as well as to word and sentence parsers and generators.

Dialogue and author systems together

At the University of Nijmegen we are working on a preliminary implementation of such a design. Through the dialogue components, the user can consult domain knowledge relevant to the goals and procedures of the office. One such domain might concern stored text documents which have been processed (written, edited, criticised, authorised, mailed, received, translated, updated, etc) by office workers. A typical user question might be: 'Did I finish and mail the letter to Mrs X?'. After having located and retrieved the document, the dialogue system hands it over to the author system, which proceeds by displaying the text in a window on the screen.

An important assumption underlying the design of such a system is that the author system can treat a text both as a sequence of characters—stored in a file, displayed on the screen—and as a linguistically structured object. Both representations of the text are maintained and operated upon interdependently.

For example, suppose the user instructs the author system to put a passive sentence into active voice (by issuing a command to the tree editor whose job it is to manipulate linguistic structures, in particular syntactic trees). The author system then responds by modifying not only the linguistic structure of the sentence but also its image in the text file and on the screen. On the other hand, when the user modifies the screen image of a sentence directly, for example by inserting the plural ending of a noun, then the author system will automatically adapt the corresponding linguistic structure and carry out any implied alterations to other parts of the sentence.

Text editing

Thus users have two ways of editing a text. They can directly modify its orthographic representation by typing into a displayed text file—this is the procedure followed in present day word processors. In addition, they can call the tree editor and propose alterations to the underlying structural (linguistic) representation. In the former case, the parser components will take care of adapting the corresponding linguistic structures. In the latter case, the word and sentence generator will reconfigure the linguistic structure. Both procedures will cause text file and screen image to contain not only the writer's explicit edits but also the ones that are entailed on linguistic grounds.

It seems to me that the kind of text editing facilities described should be part of any office workstation that truly deserves the epithet of 'fifth generation'. I also hold the opinion that the proposed technique of text editing on the basis of two interlinked text representations—one orthographic, one linguistic—departs considerably from even the most advanced word processing techniques to date. This is a second reason for giving 'author systems' the 'fifth generation' epithet: namely as the successor to four generations of handwriting, printing, typewriting and word processing.

Gerard Kempen ●