THE PRAGUE BULLETIN OF MATHEMATICAL LINGUISTICS 47, 1987

"SPSS" - An Algorithm and Data Structures Design for a Machine Aided English-to-Czech Translation System

Petr Strossa

1. Introduction

As it was stated many times (cf., e.g., Vauquois, 1975), it is practically impossible today to aspire to a universal, high quality fully automatic system for translation between two natural languages. Even if we restrict our interest to translation of texts from a limited domain with relatively stable terminology (e.g., texts about electronics or pumping technology), there are still huge problems to deal with, and the fully automatic translation systems are at least very much memory and time consuming and/or they are not able to translate all the sentences of the input texts, so they require post-editing as well. The main problems are due to unsolvable or unrecovered ambiguity of some words and syntactic constructions. Such an ambiguity can be unsolvable in principle without special knowledge included in the translation process (cf. the example by Revzin and Rozentsveig - Russian "syn tsarya Fyodora" says in English either "Tzar Fyodor's son" or "a son of Tzar Fyodor", depending on how many sons Tzar Fyodor had, if he existed at all - and we can assume any human translator to be able to find this information, or it may stay unrecovered simply because the system is not adapted to treat the word or construction as ambiguous. In the first case the system wastes time by creating "possible" translat-

ions of a sentence which would be quickly rejected by any human translator considering his knowledge of the domain and the context (including the suprasentential one, dealing with which brings further problems in automatic systems), while in the second case we obtain a partially invalid translation without any indication, although a human translator could say some of the output sentences to be nonsense if he had seen them before their including into the output text. Furthermore, there comes the problem of how is an automatic system to treat the words and constructions that it does not know because its authors did not expect their occurrence in processed texts.

Rather than fully automatic systems including knowledge bases, interactive systems are now considered hopeful in overcoming the described problems with acceptable demands as to human work and machine capacities. We can say that the possibilities of interactivity in machine translation are searched into for just as long a time as the possibilities of interactivity in computing in general. One way is described already by Vauquois (1975): he proposes an interactive superstructure to an automatic system of translation with syntactic analysis. Such a superstructure would enable the user to enter into the process with his intelligence whenever it could not resolve a question automatically. However, it would require a user relatively well informed about the way the system works "inside". Furthermore, such a solution looks like a "human-aided machine process", rather than machine aided human work.

There are two principal points of view giving reason to *machine aided translation* (henceforth *MAT*) systems as they are referred to, e.g., by Tanke (1979) and Melby (1982). Firstly: most of the time spent by translators of scientific and technical texts in their work is filled up with looking into dictionaries, and in spite of this they often cannot be sure as to their results - because of temporal changes of terminological vocabulary. Secondly: as it was already said above, if one wants to use a computer he wants to work with the aid of the computer and not to help the computer in its work,

i.e., the user wants to select himself the extent to which the work is to be done automatically and to which the responsibility is to be kept by him.

Melby proposed, and brought as far as to series production later, a MAT system for a microcomputer with three main levels of automatization of the process. At the first level the system works as a text processor (i.e., it provides for functions such as creating, scrolling and various editing of texts in natural language) together with an interactive dictionary (that gives dictionary entries to the words keyed by the user according to his need). At the second level the system processes an input text automatically and offers possible translations of words and phrases to the user who is still keeping full responsibility for the form of the output. The third level executes programs for automatic translation of the whole text, but marks every sentence which is likely to be translated doubtfully, so that the user could easily find these sentences in the output and revise them before printing. The user of Melby's system can select the level of automation before the whole work as well as change it any time during the work when he sees that the previously selected level was not satisfying.

In my diploma work elaborated at the department of applied mathematics of the Faculty of Mathematics and Physics, Charles University, Prague, under the supervision of Eva Hajičová, CSc., I wanted to design an algorithm and data structures for a machine aided English-to-Czech translation system. To simplify the task I presumed two levels of automation - the first two described above - and I excluded the text processor functions from the focus of my interest, taking into account that a subsystem with such functions could be added later. So my study shows the linguistic problems of such a MAT system and possibilities of their solution.

2. The task from the linguistic point of view

At both levels of automation, the first one (let us call it *dictionary mode*) as well as the second one (let us call it *automatic input text processing* or *AITP mode*), the essence of the task is to retrieve dictionary entries according to words specified in some way. But even in the dictionary mode the user should not be forced to type his inputs in basic (dictionary) forms: he may not know what the correct basic form of a word found in a text is, either; he may not know, e.g., whether the verb, from which "interpreted" is derived, is "to interpret" or "to interprete" (misspelled and misread to rhyme with "complete"). And in the AITP mode, it is evident that the system must be able to accept words in all their grammatical forms, as they occur in normal texts. It would be surely wasting of memory to have a special dictionary entry for every regularly derived grammatical form in English, e.g., for every -ed-form, -ing-form, regular plural, etc. But once we have a procedure for recovering basic forms of the input words (let us call it *lemmatization procedure*) in the system, we can use it for more than that: such a procedure can be adapted to recognize various words as derived from other ones, or as probably derived from other ones, if they are not found themselves in the dictionary. Thus we could automatically obtain the dictionary entry of "develop" as the answer to "development", as well as that of "compute" to "computer" or "computation", etc. However, it would not be useful to include the less frequent word-formation segments into the lemmatization procedure, as this could unnecessarily slow down the work of the system with every word. Anyway, it is necessary first to look up every word in the dictionary before trying to re-derive its basic form, as a special form can always have its special meaning (cf. "means" and "mean", "foundation" in the meaning of "fund" which cannot be derived from the equivalent of "found" in other languages). Then we must realize that the process

of lemmatization is generally indeterministic. We cannot know, e.g., whether "lives", treated as an isolated word, is a form of "life" or "live", as well as we cannot tell "bases" as plural of "base" from that of "basis" without taking into account the semantic context. Furthermore, it is useful to keep some information about the original input form (or some information implied by it) until giving the whole answer to the user, so that we could avoid wrong recognitions such as the noun "swallow" in the form "swallowed". Naturally the system must also deal with the irregular forms of words in a way acceptable for the user.

A whole class of problems appears in the AITP mode. They are namely:

- *recognizing of idioms and composed terms in the text:* idioms and some composed terms are not translatable word-by--word; if there is such a construction in the input text, the system must answer by a translation of the whole construction and should not offer a translation of the single words constituting it, as they are not interesting for the user in this case (while in the dictionary mode the user probably will be interested both in the word as such and in all the idioms and terms containing it).

- *modifiable verbs:* we use this term for all English verbs the meaning of which can be considerably modified by prepositions or short adverbs; cf. "deal" and "deal with", "give" and "give up" etc.; these prepositions and adverbs (called *verbal modifiers* in my system) can be placed in the sentence in a theoretically unlimited distance from the verb they modify (sentences like "he *deals* primarily, but not exclusively, *with* theoretical problems" or "he *gives* his functions gradually *up*" are quite permissible, though maybe not quite usual) and the system must be able to connect the verb with its modifier (if there is any in the sentence) before giving any dictionary entry to the user; in the dictionary mode, on the other hand, we consider useful to inform the user that a specified verb is modifiable so that he could add a modifier to it in his next query.

- *"incomplete idioms"* (in Vauquois' terminology) like "fit (something) in place" or "take to (one's) heels": as a matter of fact, in usual dictionaries these idioms represent only skeletons with "parameters" (here in parentheses) which are replaced by variable length expressions in the "instances" of the skeletons in texts; the system must recognize the two parts of such a skeleton separated by some expression in the text and treat them together is any idiom (see above).

- *hyphen constructions:* the use of hyphen in English is often relatively free; we can see, e.g., "high-pressure pump" as well as "high pressure pump","over-view" or "super-structure" as well as "overview" or "superstructure", etc.; thus the system should be able to treat two words connected with a hyphen either as one word (with the hyphen left in place or omitted) or as two separate words so as to enable the correct matching of dictionaries and input texts with various orthographic usages.

- *negative words:* in a MAT system we use this term for words whose translation we do not assume to be interesting for the user - grammatical words as articles, auxiliary verbs, numerals etc.; often repeated retrieval of dictionary entries to these words would make the system clumsy and uncomfortable; such words should be recognized very quickly and overridden without any actions more; i.e., there must be a quick test in the system that resolves the question whether the current word is or is not negative.

3. The data structures and the principles of the algorithm

There are 3 dictionaries in "SPSS" ("Systém překladu za pomoci stroje", i.e. "Machine Aided Translation System"): the first one, called *Basic Dictionary (BD)*, contains single English nouns, adjectives, verbs and abbreviations with their necessary grammatical data and Czech equivalents or logical pointers to other dictionary entries (see below); the second one, called *Dictionary of Idioms and Composed Terms (DICT)*

contains English idioms and composed terms with their Czech equivalents; the third one is the *Negative Dictionary (ND)*, containing negative words (see part 2).

It must be pointed out that both BD and DICT require direct access memory device and we have to assume them to be relatively large and incomplete at every moment in the practical use of such a system. ("SPSS" uses 57 bytes for a BD entry and 153 bytes for a DICT one - and we could take account of about 10,000 entries for BD and even more for DICT in a commercial system for some technical domain). In the experimental version of "SPPS", BD and DICT are represented by indexed sequential disk files, but this is not to say that another direct access organization could not be better. On the other hand, ND is relatively small and can be stored in the working memory during the work of the system. (In the experimental version ND consists of 151 entries representing common English pronouns, possessive and numeral adjectives, articles, auxiliary verbs, primitive adverbs, prepositions and conjunctions, and although this realization of ND is not really complete, ND can be treated as complete - negative words are not productive and a complete list including a few hundreds of them could be created for a practical application of the system.) For the representation of ND in the memory, I use a hashing table with 225 fields, each of them being either empty or pointing at a linked list of negative words (see Fig. 1).

Fig. 2 shows 4 main types of (abstract) BD entries. The pair' *WORD+SEQ.NO.* (occurrence sequence number of an English word in the dictionary) forms the logical access key of an entry (which must be transformed to the necessary physical form in every concrete implementation, e.g., to one string of characters in the standard ISAM used). The "LEFT 1" entry is the type pointing (by its *TARGET*) at a DICT entry with *EXPR.* containing the word "LEFT" ("STATION 1" being its logical access key; it can be seen that in DICT this key can be determined arbitrarily for every entry, as DICT is accessed only

through the BD entries of the type described here). The *TARGET* of "LEFT 2" points at all the BD entries with *WORD* = "LEAVE": it is an example of an irregular form or word abbreviation entry. "LEFT 3" is a normal entry with Czech equivalent of the adjective "LEFT" in the *TARGET*. "DEAL 1" is an example of so-called *modificable verb foreshadowing*, with some values undefined, characterized by a value greater than 4000 in a specified technical data item: it only points at the fact that there is at least 1 entry with *WORD* like "DEAL IN", "DEAL WITH" or similar somewhere in BD.

Now let us demonstrate the function of the system in the AITP mode. The system takes the input text sentence-by-sentence and every sentence word-by-word. At every word it first tests whether it is not a negative word: it computes the respective hashing function value and looks for the current word in ND. If it is found there, it is overridden; otherwise a lookup is made in BD for the entries identified by (W,1), (W,2), ... (while there are such), where W is the current word. If some entry found is of the 1st type described above, a subroutine for matching composed expressions is called (which can deal with "incomplete idioms", too); if it is of the 2nd type, a new BD lookup is made recursively; if it is of the 4th type, a verb modifier is looked for until the end of the sentence, and if it is found, a new BD lookup is made for the modified verb. (As all the verb modifiers are either prepositions or primitive adverbs, they are marked in ND, which is thus used at this step, too.) If (W,1) is not found, a lemmatization is tried. If this is still unsuccessful, a hyphen is looked for in the word and if found, new lookups are made for the two parts (the attempt to transform a word with a hyphen to one word without it is not included, but it would be a matter of a very simple modification).

It should be mentioned here that the BD entries of the same English word are ordered according to their technical data so that modifiable verb foreshadowings come first, followed by pointers to DICT, and other entries come at the end.

Thus the first two problems referred to in the 2nd paragraph of part 2 are solved in the AITP mode. In the dictionary mode, the process with every word is almost the same, except that all the entries (W,1), (W,2),..., (W,n) are first pushed down into a special pushdown store and then processed in reversed order, which satisfies the requirements for the form of the output in this mode. The two modes are "memory-balanced" as one needs some extra memory for the pushdown store, while the other for storing the current sentence.

An important part of SPPS is the lemmatization procedure. Its algorithm is briefly illustrated by a kind of transition table for English word forms ending in "-s" in Fig. 3. We can realize such a table for every possible end character of English word forms and word formation suffixes (although our table covers only the "-s" of plural and 3rd person). Column I of the table contains labels of rows representing possible replacings and their conditions; S1 is the entry point. Replaced end segments are in column II, replacing ones in column III: if the end segment from column II is found in the processed word form, the replacing is applied and followed by a BD lookup; now column IV shows what can be the value of a specified item in the BD entry (e.g., 100 means "noun", 300 means "verb never doubling its end character", 350 means "verb doubling its end character in some forms" - these categories were adopted pragmatically, according to the class of functions of "SPPS"). If the replacing has not been applied, the next row of the table is tried, while if the dictionary lookup has not been successful, a new row to be tried (if there is any) is found in column V; this row is tried even if the last dictionary lookup has been successful, supposed there is a plus sign in column VI (thus the ambiguity of forms like "lives" or "axes" is dealt with). The global plan of processing a word form using a set of such tables is as follows: (i) the respective table is selected according to the end character of the word form; (ii) the control is taken over by the table; (iii) if no success

has been achieved, an attempt is made to remove a prefix of a predetermined class (e.g., "un-" or "non-" in the experimental version) and then to repeat the process from (i).

4. Conclusion

"SPPS" exists and is being debugged as a program in PL'I-Optimizer programming language for a big computer (EC 1040, 1045). The system works with debugging BD and DICT with about 1000 and 100 entries, respectively, The program has about 2500 lines of source text and works in about 200 K bytes memory. Its present version is not really interactive, it only simulates interactivity, having all inputs punched on cards (or prepared on a disk or tape) and outputs printed. Thus "SPPS" is far enough from a real machine aid for English-to-Czech translation. However, "SPPS" shows what main problems must be solved by such an aid and how they can be solved. Some ways of possible improvements of this system have been mentioned in my diploma work, and further development of "SPPS" to a real interactive aid is envisaged.

References

Melby, A.K. (1982), Multi-Level Translation Aids in a Distributed System; in: COLING 82, Proceedings of the Ninth International Conference of Computational Linguistics, Amsterdam, North Holland - Academia, Prague

Nagao, M. (1983), Summary Report on Machine Translation, PBML 39.

Tanke, E. H. (1979), Implementing Machine Aids to Translation; in: Snell, B. M. (ed.): Translating and the Computer, Amsterdam, North Holland

Vauquois, B. (1975): La Traduction Automatique a Grenoble, Documents de Linguistique Quantitative, 24, Paris, Dunod

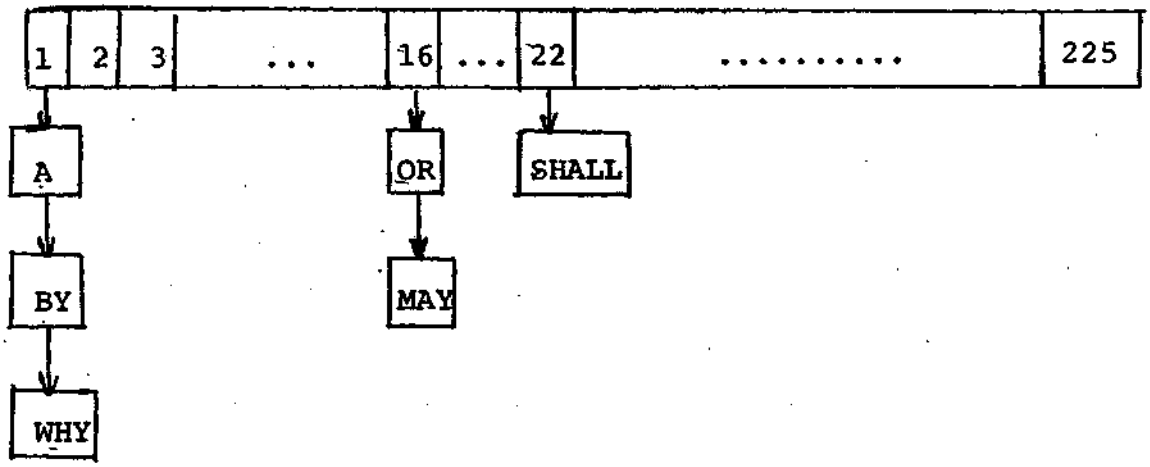

Fig. 1. The scheme of the negative dictionary stored in the working memory

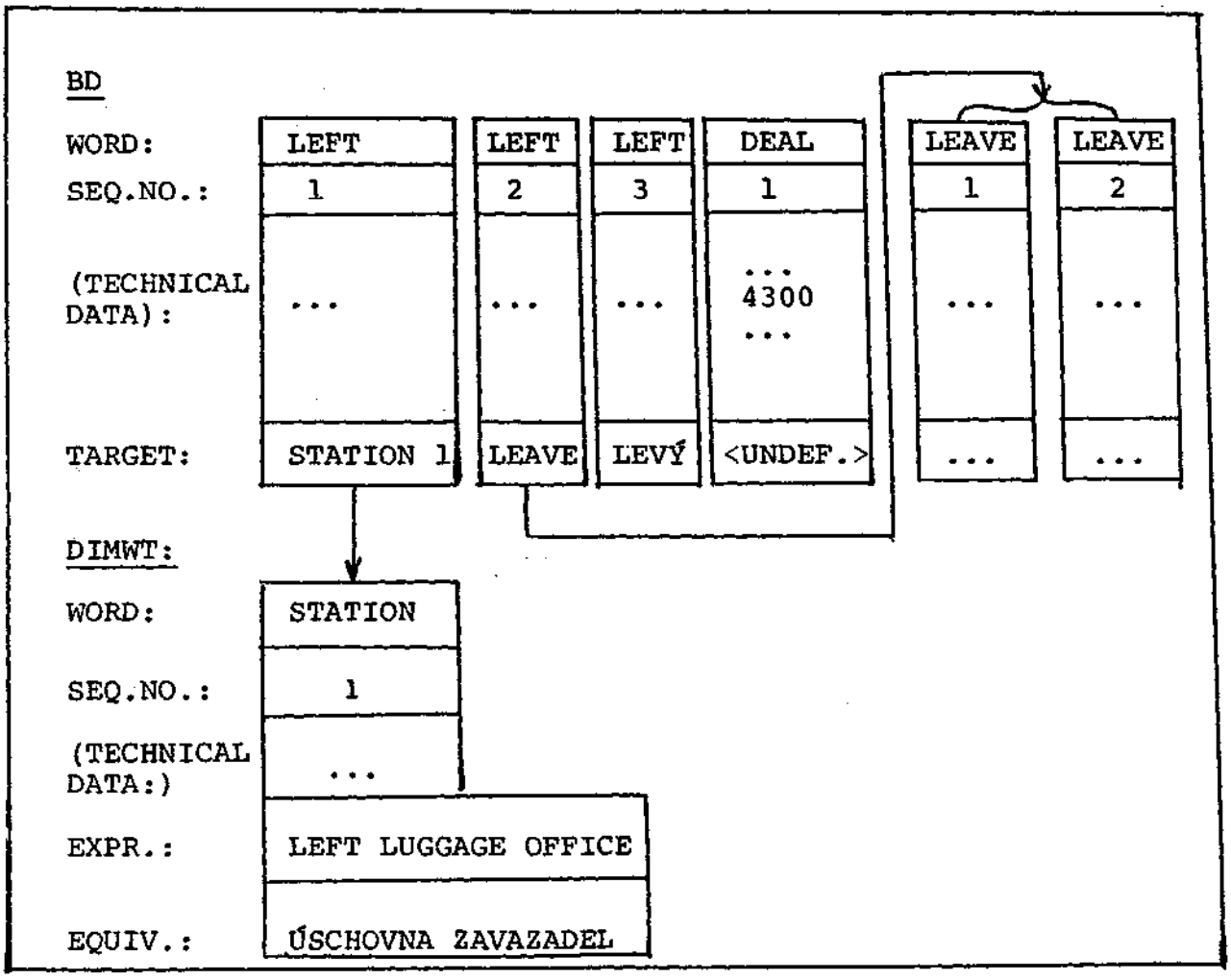

Fig. 2: The scheme of BD and DIMWT entries and their relations

I	II	III	IV	V	VI
S1	ses	s	100, 300	S10	+
S2	xes	x		S10	+
S3	zes	z		S11	
S4	shes	sh		no	
S5	oes	o		S11	
S7	ves	f		S11	+
S8	ies	y		S11	+
S9	ices	ex	100	S11	
S10	es	is	100	S11	+
S11	s	no	100, 300, 350	no	

Fig. 3: The re-derivation table for word forms ending in "-s".