scheme used by many developers of stemming algorithms. This scheme was first represented completely and in full detail in the widely known algorithm constructed by J. B. Lovins, which was publicized in 1968 (Lovins, 1968). For instance, she divided all suffixes into three groups (A, B, C) and provided the list of all three groups. Lovins introduced the following rules: (1) any suffix from group A can be removed from the word without paying attention to the number of letters in the remaining part of the word (stem); (2) if a suffix from group B is removed, then the minimal remaining stem must be three letters long; (3) suffixes from group C can be removed only when the minimal remaining stem contains four letters. Lovins also introduced separate rules for several indexes. In particular, she specified that the suffix YL permits removal only after letters N or R, the suffix ING is removed in all cases unless the remaining word after deletion consists only of one letter or the letters TH, the suffix ALLIC specifies a minimum stem length of three (group B) and prevents suffix removal after MET and RYST; and so forth.

In addition, the so-called transformation rules are frequently used in the framework of stem separation. For example, the word ABSORB is transformed into ABSORPTION when the suffix TION is added. Special rules are introduced (they exist in the Lovins algorithm as well) to provide for the substitution of the letter P by the letter B when the suffix TION is removed. However, these rules, which are important for translation in natural languages, are not so essential for the purposes of indexing, because the main parts of words included in conditional equivalence classes should not necessarily be readable or grammatically correct. Moreover, the same word may have more than one stem.

Obviously, the variety of rules that Lovins proposed (we have described only some of them) do not cover all the linguistic situations that can arise in the recognition of words. Mainly this is due to a certain percentage of errors in the stem separation provided by her algorithm. It is worth noting that the literature in this area reveals that one of the main ways chosen by the developers of stemming algorithms to reduce errors is to detect cases that do not satisfy the existing rules and then to introduce new rules (frequently by expanding existing ones) that will account for such cases. Such new rules work until new situations that break these rules are found. Then developers must make new additions to the previous additions and so forth. At the same time, Meadow's opinion is that "A workable stemming program would probably require at least 10–20 rules (hundreds are possible), including large numbers of provisions for special cases and irregular words" (Meadows, 1992).

Other authors have also indicated that indexing does not necessarily require precise stemming algorithms. This seems to be so, because the introduction of any new rule makes the algorithm more awkward, degrading its performance and complicating its implementation. Moreover, a number of IR system properties compensate for errors occurring during the stemming procedure. Indeed, suppose a document contains five different word forms of a certain term

and during stemming only one of them is correctly recognized and separated. This single recognition is sufficient for including the corresponding descriptor in the document profile. Even if the algorithm fails to recognize all five word forms and the corresponding descriptor is not included in the document profile, there is still a certain probability that the document will probably still be correctly retrieved owing to other descriptors in the document profile. In any case, many information retrieval experts do not see any problems with stemming. For instance, Salton and McGill reasoned, "It is not difficult to implement a suffix removal algorithm producing usable word stems for the vast majority of existing English word forms" (Salton & McGill, 1983).

We noted earlier that stemming algorithms in IR systems are used to alleviate the problem of recognizing words in the course of automatic indexing. However, using these algorithms is not a necessity. The fact that the very text of the document can be used as a document profile, eliminating any need for indexing procedures, is not the only reason for that. Even when automatic indexing is implemented, stems are not always identified. Another way to recognize different word forms exists. Since the 1960s, IR system developers understood that the simplest way to eliminate all the difficulties concerning the recognition of words was to include all possible word forms into the conditional equivalence classes containing these words. However, for the English language this would enlarge the descriptor dictionary 10- to 12-fold. Because at that time computer memory was rather restricted, this method was judged technically unacceptable. However, the technical capabilities of modern computers have allowed developers to implement this idea as an alternative to stem separation. Thus, the fact that the majority of modern systems use stemming algorithms is due more to inertia rather than to reasonable arguments. In any case, the alternative seems not only simpler and more convenient for its practical implementation but also more reliable and stable. Indeed, one can easily see that in this case automatic indexing constitutes merely a word-to-word comparison of words from the text being indexed with words from the descriptor dictionary.

Among other problems encountered in the development of automatic indexing algorithms, lexical homonyms (developers try to make the algorithm capable of recognizing the correct meaning of homonyms) and word combinations (an attempt to elucidate stable word combinations) are mentioned most frequently. It is commonly noted that solving these problems will reduce noise in the output. In principle, practically all researchers agree with that, although a number of experiments have indicated that in the vast majority of cases the effects of elucidating word forms and particularly of distinguishing homonyms are virtually negligible. Nevertheless, a brief description of these problems and the most typical methods solving them will be useful. Let us start with methods for distinguishing the meanings of homonyms.

In everyday speech, we recognize the meaning of a homonym through its surroundings, that is, in context. Therefore, when developing automatic meth-