Automatic Language-Specific Stemming in Information Retrieval

John A. Goldsmith[1], Derrick Higgins[2], and Svetlana Soglasnova[3]

Department of Linguistics, University of Chicago, 1010 E. 59th St., Chicago IL 60637 USA
[1]ja-goldsmith@uchicago.edu, [2]dchiggin@midway.uchicago.edu,
[3]s-soglasnova@uchicago.edu

Abstract. We employ *Automorphology*, an MDL-based algorithm that determines the suffixes present in a language-sample with no prior knowledge of the language in question, and describe our experiments on the usefulness of this approach for Information Retrieval, employing this stemmer in a SMART-based IR engine.

1 Introduction

The research discussed in this volume is directed at the special character of Information Retrieval in the multilingual world which is the future of the information age. What special challenges must we be ready for as we prepare our document bases and document spaces for texts in a potentially unlimited number of languages? What additional technology must we develop in preparation for those challenges?[1]

To the extent that current IR methods make assumptions about language which are valid for English but not for many other natural languages, these methods will need to be updated in the light of what we know about natural languages more generally. Our concern in the work reported here is the need for stemming (and related processes) that is fast, accurate, valid for as many languages as possible, and that assumes no human intervention in the process.

We are currently in the process of developing software that accepts unrestricted corpora as input and produces, as its output, a list of stems and affixes found in the corpus, plus additional information about cooccurrence of affix and stem. It does this on the basis of no prior knowledge of the language found in the corpus. When linked to an automatic language identification system, such a system is able to add to our ability to control a large document base which must accept documents in any language—such as the Internet, for example. Although the testing done in the context of the CLEF experiments deals with some of the larger European languages, we see our approach as being most useful when it is used in relatio to a database that includes a large number of documents from little-studied languages, because morphologies cannot be produced overnight by humans.

[1] We are grateful for help and comments from Abraham Bookstein and Craig Swietlik. This work was supported in part by a grant from the Univeristy of Chicago-Argonne National Laboratory.

Our background is in linguistics and computational linguistics, rather than information retrieval (IR), but in the next section we will survey what we take to be the relevant background information regarding the character of stemming for IR in English and other languages.

2 Multilingual Stemming

The use of stemming in information retrieval systems is widespread, though not entirely uncontroversial. It is used primarily for query-stemming and document indexing. (Useful reviews may be found in [2], [11], [13].).

Stemming in the narrowest sense is "a process that strips off affixes and leaves you with a stem" [20:132]. A broader procedure is *conflation:* "a computational procedure which identifies word variants and reduces them to a single canonical form" [17:177]. Word variants are usually morphological [2:131] or semantical [23:633]. Stemming in the narrow sense is a type of conflation procedure. Very commonly, though, the term is used not just in that narrow sense, but to refer to lemmatization [12:654], or collapsing [17]. "Stemming" in query expansion refers to that second sense. For our purposes, *stemming* is taken in a broad, but not the broadest, sense. Any algorithm that results in segmenting a word into stem and affixes is a stemming algorithm, or stemmer.

Significant factors for stemming performance in IR include the type of stemming algorithm, evaluation measures of retrieval success, language-(in)dependence, query length, document length, and possibly others [15]. These issues have been addressed in many studies, but no clear comprehensive picture emerges from the literature.

By its very nature, stemming is generally understood to improve recall, but to decrease precision [29:124]. Most research on stemming in IR is on English, a language with a relatively simple morphology. In a study comparing three different stemmers of English, Harman [9] found that losses in precision from stemming outweigh the benefits from increased recall. Krovetz [16] reported results conflicting with what Harman found for the Porter algorithm on the same collection using a very close evaluation measure [15], and in general the view that overall stemming is beneficial for IR is discussed in [28:6], [13], [2], and [17].

2.1 Types of stemmers and evaluation measures

Stemmers may be *linguistic*, *automatic* or *mixed*. Linguistic stemmers use a linguist's knowledge of the structure of the language in one way or another, typically by providing manually compiled lists of suffixes, allomorphy rules, and the like. The best known stemmer of this sort is Porter [26], initially developed for English. Porter's approach was extended to French and Italian [30] and Dutch [15]. Automatic stemmers rely on statistical procedures, such as frequency count, n-gram method, or some combination of these. Linguistic stemmers that rely on statistical methods as subsidiary procedures may be called mixed. Such mixed system include [16] and [23]. Krovetz [16] uses frequency of English derivational endings as the basis for incorporating them into the stemmer, and the initial shared trigram as a preliminary procedure for finding words that are potentially morphologically related. Paice [23] requires the words in a manually compiled semantic identity class to share the initial bigram.

It has been pointed out in the literature that it is difficult to evaluate and compare the performance of different stemming algorithms for IR purposes because the traditional IR evaluation measures are not aimed at highlighting the contribution of stemming to query success [10],[11],[16],[23]. Several studies that compare the effectiveness of different stemming algorithms for IR [9],[10],[16],[17],[23] were conducted on English materials, with Paice [23] and Hull [10] developing new measures of evaluating stemming performance for IR. The results are inconclusive.

Lennon et al. [17] evaluated seven stemming algorithms for English for their usefulness in IR. The automatic algorithms in this study were the RADCOL [19], Hafer-Weiss [8], a similarity stemmer developed by the authors on the basis of Adamson and Boreham's bigram stemmer [1], and a frequency algorithm developed by the authors on the basis of RADCOL. The linguistic stemmers were Lovins and Porter. The Hafer-Weiss algorithm fared much worse than all others. With this exception, they found an undeniable, but very slight improvement on stemmed queried compared to unstemmed ones. They also found "no relationship between the strength of an algorithm and the consequent retrieval effectiveness arising from its use".

Harman [9] tested three linguistic stemmers: Porter, SMART-enhanced Lovins stemmer, and the primitive s-stripping stemmer for IR effectiveness. She found that the minimal s-stemming did very little to improve IR effectiveness, and more rich stemming hurts precision as much as it improves the recall.

Hull [10] evaluated five linguistic stemmers for English: *s*-remover, an extensively modified Lovins stemmer, Porter stemmer, Xerox English inflectional analyzer and Xerox English derivational analyzer. He proposed a set of alternative evaluation measures aimed to distinguish performance details of various stemmers. In his analysis, stemming is much more helpful on short queries, on which the inflectional stemmer looks slightly less effective, and the Porter stemmer slightly better, than the others; the simple plural removal is less effective than more complex stemmers, but quite competitive when only a small number of documents is examined. His detailed analysis of queries shows how linguistic knowledge may be beneficial for IR in some cases (*failure/fail*—only the derivational stemmer makes this connection) but not in others (*optics/optic—the* derivational and inflectional stemmers do not make this connection).

Paice [23] developed a direct measure of evaluating accuracy of a stemmer "by counting the actual understemming and overstemming errors which it commits". He evaluated three stemmers for the English language— Porter, Lovins and Paice/Husk [24]. It was found that his measure provides a good representation of stemmer weight, but no clear comparison of accuracy for stemmers differing greatly in weight. There is no clear relationship between IR measures and Paice's evaluation.

The upshot appears to be that for English, the choice of stemmer type ultimately does not matter much (though cf. [3]). Krovetz [16] found that his inflectional stemmer always helped a little, but the important improvement came from his derivational stemmer. Lennon et al. [17] and Hull [10] found no overall consistent differences between stemming algorithms of various types, though on a particular query one algorithm might outperform other, but never consistently. Most studies note that stemming performance varies on different collections. Paice [22] notes that heavy stemmers might be preferable in situations where high recall is needed, and lighter stemmers where precision is more important.

For languages with morphology richer than that of English, differences between inflectional and derivational morphology—and, consequently, between performance of stemmers oriented towards one or the other—should be greater. Stripping off inflectional morphology should result in more than slight recall improvement without significantly hurting precision. In Russian, for example, the nominal declension has two numbers and six cases (declension paradigms are determined by the gender of the noun and the phonological form of the stem). Dictionary entries are listed in the nominative singular, and one would expect most queries to be entered in the "dictionary form"—the nominative singular. However, actual occurrences of the word appearing in the texts could be more frequent in oblique cases and in the plural. For example, a search for the nominative singular of the word *ruka* 'hand' in Leo Tolstoy's *Anna Karenina* (over 345,000 words) would locate 18 occurrences of the exact match. The stem *ruk*, on the other hand, appears 690 times—in forms inflected for case and number. Most frequent forms are *ruk-u* (accusative singular) and *ruk* (genitive singular, nominative plural). Nozhov [21,22] reports that all Russian IR system routinely use stemming (linguistic or mixed) even when the degree of morphological recognition is not extremely high.

Kraaij and Pohlmann [15] compared the Porter-style algorithm they implemented for Dutch, another morphologically complex language, with their more linguistically sophisticated derivational and inflectional stemmers. The best performance was achieved by the inflectional stemming combined with a sophisticated version of compound splitting and generating. Applying both derivational and inflectional stemming generally reduces precision too much.

Wexler et al. [30] developed a four-language search engine (French, Italian, German and English) with stemming implemented for each language. For German, a language morphologically close to Dutch, they apparently implemented some inflectional stemming and a dictionary-based compound-breaking algorithm.

A derivational stemmer could produce a theoretically irreproachable result which is not just irrelevant, but harmful for IR purposes, since the stem and its derivates are rarely fully synonymous. The problem is to distinguish derivation that preserves word sense relevant to the query from the derivation that does not. Hull's study gives examples of the derivational stemmer outperforming others on queries like *bank failures* (*failure* converted to *fail*), and *superconductivity* (stem *superconduct* conflated with the one in *superconductors*). Since the relevant documents contained both *failure* and *fail*, and *superconductors* rather than *superconductivity*, the stemming was beneficial. However, in cases like *client-server architecture* (conflate with *serve*) and *Productivity Statistics for the U.S.Economy* (conflate with *produce*) the linguistically correct analysis lowers precision dramatically, since *serve* and *produce* have a much less specific meaning than the query term. The lexical equivalence requirement may be maintained through manually compiled lists ([23], [16] for English), or by word sense disambiguation in a full-blown NLP system ([11] for French).

2.2 Automatic stemmer on more than one language

The increasingly multi-language character of IR [7] presents a special challenge to language-specific tools. Statistical language processing tools, with their universality and speed, are understandably attractive in this regard. Whether stemming based on such universal methods helps to increase accuracy and scope of IR is a question without a definitive answer yet.

Xu and Croft [31] tested the performance of an automatic trigram stemmer, a "general-purpose language tool" against the performance of Porter stemmer and KStem [16] on English and Spanish corpora for construction of "initial equivalence classes". The initial equivalence classes were further refined with statistical methods that differed for English and Spanish. The "trigram approach" was used as an auxiliary procedure to clean up the equivalence classes for English after the application of the connected component algorithm: A "prefix" in an equivalence class is defined as "an initial character string shared by more than 100 words. Examples are *con*, *com* and *inter*. If the next 3 characters after the common prefix do not match, the similarity metric is set to 0. Thus, the trigram model is at work again, shifted further inside the string. The results were comparable with the performance of the linguistic Porter and KStem stemmers, showing some portability problems due to corpus-specific character of equivalence classes.

2.3 Compounds

As virtually all studies on IR in German have documented (and as reported in this year's CLEF results by the West Group; see also [15]), it is crucial to analyze compound words in German, and no doubt in other languages with similar use of compound structures. Use of automatic morphology can be of significant help in this area, as reported in [6] in connection with Automorphology. Because our algorithm identifies stems, it is possible to identify compounds, which take the form Stem-Linker-Stem-Suffix; that is, the first half of the compound need not be a free-standing word.

3 Automatic Morphology

The identification of a lexical stem consists of the identification of a string of letters which co-occurs in a large corpus with several distinct suffixes, and typically we will find consistent sets of suffixes that appear with a wide range of stems. This observation serves as one of the bases for our algorithm, whose goal is to establish as wide a range of stems and suffix possibilities as possible, given a corpus from a natural language. The following discussion is a summary of material presented in [4],[5]. Its goal is to establish a method which is language-independent, to the extent possible, and which will provide a useful result despite the lack of any human oversight by a speaker of the language in question.

There are several methods that can be used to establish an initial set of candidate suffixes on a statistical basis, given a sample of an unknown language. One of the simplest is to consider all word-final sequences of six or fewer letters (*schaft* is a German suffix), and to rank their *coherence* in the text on the basis of the formula in (1). In order to deal appropriately with single-letter suffixes, it is preferable to consider all words to end with a special symbol, and to increase the maximum size to seven letters. The frequency of a letter is defined as the number of occurrences of the letter in the text divided by the total number of letters in the text.

$$freq(l_1 l_2 ... l_n) \log \frac{freq(l_1 l_2 ... l_n)}{freq(l_1)\, freq(l_2) ...\, freq(l_n)} \tag{1}$$

We select the top 100 suffixes ranked by coherence (1) (these are our *candidate suffixes*), and divide all words into stem and suffix if they end in one or more candidate suffixes. We associate with each such candidate stem the set of suffixes it occurs with, and call each such set a *candidate signature*. We accept only signatures with at least two suffixes, and we establish a threshold number of stems which a signature must be associated with, failing which a signature is eliminated; a suitable threshold is 5.

Various improvements can be made to the results at this point. For example, common combinations of suffixes are certain to be identified as suffixes (e.g., *ments, ings* in English), but they can be identified and their stems reanalyzed. A large part of our work is devoted to determining in an abstract way what kinds of errors our algorithms are likely to create, to determine what they are, and to find ways either to avoid the errors or to undo them after the fact, but always without human intervention. Our current system is heavily based on a Minimum Description Length analysis [26], one consequence of which is that if a language has an unusually high frequency of occurrence of a specific letter in stem-final position, it is likely to be misanalyzed as being part of a suffix; this is the case for *t* in English. When viewed up closely, suffix systems tend to have certain kinds of orthographic structure which derives from their history and which can confuse an automatic analyzer; for example, Romance languages contain sets of verbal suffixes which are derived historically from inflected forms of Latin *habere*, which itself has a stem-suffix structure. The suffixes *-ai,-ais, -ait*, etc., of French may in some cases wrongly be analyzed as being *-i,- is,- it*, and attached to a stem that ends in *-a*. We employ the techniques of Minimum Description Length in order to select the analysis of the complete corpus which is most compact overall and which provides the most succinct and accurate analysis of the stem/suffix distribution.

There are two notions at the heart of the MDL approach. The first is that an analysis (here, the morphological analysis of a corpus) must provide a probabilistic measure of the data; this allows us to assign an optimal compressed length to the corpus on the basis of that model, for reasons central to information theory. In this case, each word of the corpus is identified as belonging to one of a relatively limited number of stem groupings defined by the set of suffixes the stem appears with in the corpus; this grouping is called a *signature*, and each signature is associated with an empirical probability. Each word in the corpus is also associated with a stem and a suffix, and these associations are assigned an empirical probability, conditioned by the signature of the word. Each of these three probabilities (signature, stem, suffix) for each word is converted to an optimal compression length (which equals the logarithm of the reciprocal of the probability), and the sum of these optimal compression lengths is the compressed length assigned to the corpus by the morphological model, measured in bits. The shorter that total length, the better the morphology models the corpus.

The second notion at the heart of MDL is that length of the model itself can be measured in bits, and the optimal analysis of the corpus is that for which the sum of the length of the model and the compressed length of the corpus is the smallest. Our algorithm searches the space of possible analyses by considering changes to the signature set, to the affix set, and to the stem/suffix separation, evaluating and accepting each change only if the change brings about a decrease in the total description length of the (corpus + morphology).

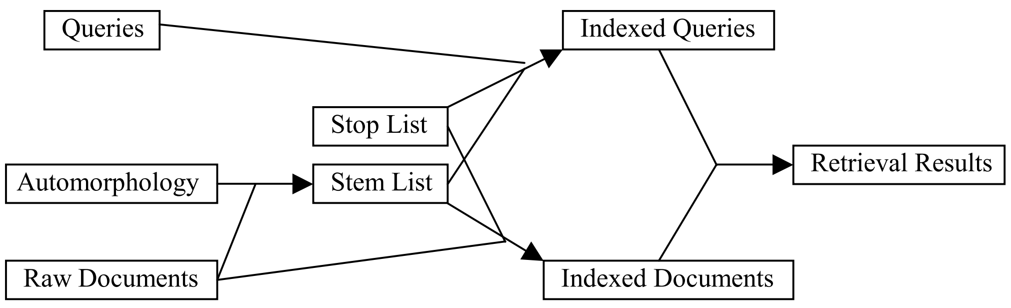

Fig. 1. The basic design of the Chicago IR system, using Automorphology to stem terms from queries and documents, and employing standard SMART vector-based retrieval.

4 Experiment

The information retrieval engine we used in our CLEF experiments is based on the freely-available SMART system, running under the Linux operating system on a commodity, off-the-shelf PC. We modified the system to incorporate our custom stemmer, which was automatically derived from the corpora for each language. The results of applying our stemmer to the document collections were stored in a file for SMART to consult at the time of indexing the documents and queries. A schematic diagram of our system architecture is presented in Figure 1. Although not represented in Figure 1, statistical compound-breaking using Automorphology was also performed on the German collection before indexing the documents and queries.

The vector-based SMART backbone is a simple retrieval models, treating each document as an unordered "bag" (i.e., retaining only frequency information), and computing document-query similarity by means of the cosine distance between these two vectors. Our expectations regarding results in this experiment were therefore guarded. Our hope is that these runs will help to highlight the strengths and weaknesses of the statistical approach to stemming for IR, and point out directions for us to progress in our development of Automorphology.

4.1 Generation of the stop and stem lists

As a stopword list for each language, we created a list of the approximately 300 highest-frequency words in a corpus of the language, and removed by hand any entries that appeared obviously inappropriate. While the resulting stop lists were by no means perfect, the lists were not long enough to create a serious problem with incorrect stopwords blocking the retrieval of documents which ought to be returned. Imperfect stoplists might, however, be blamed for not filtering out as many documents as they should, and thereby reducing our system's precision. Since our results do not seem to display a profile of high recall offset by low precision, though, the stoplists do not seem to be an area in which to look for major improvements.

The stem file for each language, which associates terms with their stem forms for indexing (a stem may be identical with the term itself), was produced by running our statistical stemming program, Automorphology, on the document collection for each language. The length of time that this process required varied from three days, for the Italian document collection, to as much as fourteen days for German, with its higher mean word length and larger document collection. Improvements in the algorithm since that work has speeded up these times considerably. The stems produced by Automorphology were accepted without any sort of human revision; the only constraint we imposed was that no stem could be shorter than three letters in length. While we do not have a concrete analysis of the conflation classes produced by our stemmer for each language, it seems likely that some of our performance deficit is due to permitting the stemmer to apply so freely.

4.2 Indexing

The indexing of documents and queries was done using standard SMART facilities, with the inclusion of the stemming routine described above into the process. Terms in document and query vectors were weighted according to the tf*idf measure which has proven effective in previous IR work. Our group used all of the permissible data fields for retrieval in each of our experiments.

Our performance on the CLEF monolingual runs might have been improved if we had invested more time in preprocessing the document collections. We did not, for example, handle issues related to diacritics at all. Thus, our system would not conflate French *Ecole* with *École*, or German *müssen* with *muessen*. However, such issues were probably not a major factor in determining the system's retrieval accuracy. Another interesting area for future exploration is the relative contribution of statistical stemming and statistical compound-breaking in indexing the German document collection. Intuitively, decompounding is less likely to do harm, since it alters terms which are less likely to be independently searched on anyhow, but it also has less potential for improvement of retrieval accuracy, because compounds are simply less frequent than non-compounds.

4.3 Retrieval

Once SMART was configured to use this new stemmer, the retrieval process for each language was straightforward. SMART uses the vector-space model to retrieve the documents most similar to the queries, using the stemmed forms of words as components of the vectors. We returned a ranked list of the top 1000 documents returned for each query, the maximum number allowed.

5 Results

Our system was run in monolingual IR tests in the CLEF project in 2000 involving Italian, French, and German. The principal results are presented in Figure 2.

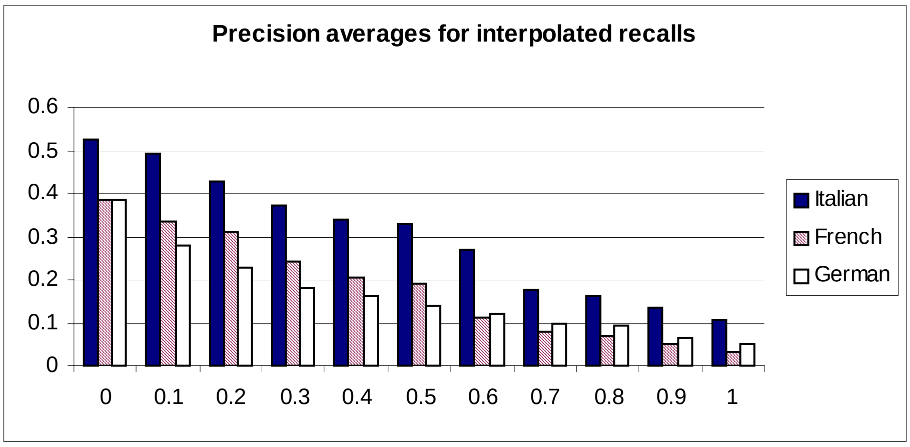

Fig. 2. Precision rates for CLEF experiments on French, German, and Italian

6 Conclusions

Our work in the area of IR is still in its preliminary stages, and we hesitate to draw any conclusions at this time from the quantitative results described here. If our work has a long-run contribution to make, it is as a component of a larger IR package, and indeed, Oard et al., in this volume, describe experiments employing our automatic morphological analyzer which in some regards goes further than our own pre-conceived ideas of its applicability. We are currently engaged in drastically reducing the time and storage needs of the algorithm to permit it to be used with databases of the magnitude typical of IR tasks, and we will continue to test the value of this work for IR tasks.

References

1. Adamson, G., Boreham, J.: The use of an association measure based on character structure to identify semantically related pairs of words and document titles. Information Storage and Retrieval 10 (1974) 253-60
2. Frakes, W.B. Stemming Algorithms. In: Frakes, W.B., Baeza-Yates, R. (eds.): Information Retrieval Data Structures and Algorithms. Prentice Hall, New Jersey (1992) 131-160
3. Fuller, M, Zobel, J.: Conflation-based Comparison of Stemming Algorithms. In: Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.
4. Goldsmith, J. Unsupervised learning of natural language morphology. To appear in *Computational Linguistics*
5. Goldsmith, J.: Linguistica: An Automatic Morphological Analyzer. In: Boyle, J., Lee, J.-H., Okrent, A. (eds.): CLS 36. Volume 1: The Main Session. Chicago Linguistic Society, Chicago (2001).
6. Goldsmith, J., Reutter, T.: Automatic Collection and Analysis of German Compounds. In: Busa, F., Mani, I., Saint-Dizier, P. (eds.): The Computational Treatment of Nominals: Proceedings of the Workshop COLING-ACL '98. COLING-ACL, Montreal (1999) 61-69.
7. Grefenstette, G. (ed.): Cross-Language Information Retrieval. Kluwer, Dordrecht (1999)
8. Hafer, M., Weiss, S: . Word segmentation by letter successor varieties. Information Storage and Retrieval 10 (1974) 371-85
9. Harman, D. How effective is suffixing? Journal of the American Society for Information Science 42 (1991) 7-15
10. Hull, D. Stemming algorithms - A case study for detailed evaluation. Journal of the American Society for Information Science 47 (1996) 70-84
11. Jacquemin, C., Tsoukermann, E. NLP for term variant extraction: synergy between morphology, lexicon, and syntax. In: Strzalkowski, T. (ed.) Natural Language Information Retrieval. Kluwer, Dordrecht (1999) 25-74
12. Jurafsky, D, Martin, J: Speech and Language Processing. Prentice Hall, Upper Saddle River NJ (2000)
13. Koskenniemi, K. Finite-state morphology and information retrieval. In: Proceedings of the ECAI-96 Workshop on Extended Finite State Models of Language ECAI, Budapest, Hungary (1996) 42-5
14. Kowalski, G. Information Retrieval Systems: Theory and Implementation. Kluwer, Dordrecht (1997)

15. Kraaij, W., Pohlmann, R. Viewing stemming as recall enhancement. In: Proceedings, 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '96) Zurich (1996) 40-48.
16. Krovetz, R. Viewing morphology as an inference process. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1993) 191-202
17. Lennon, M., Pierce, D.C., Willett, P. An evaluation of some conflation algorithms. Journal of Information Science 3 (1981) 177-183
18. Lovins, J.B. Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11 (1968) 22-31.
19. Lowe, T. C., Roberts, D.C., Kurtz, P. Additional text processing for on-line retrieval. (The RADCOL System). Tech.Rep. RADC-TR-73-337 (1973)
20. Manning, C., Schütze, H: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge MA (1999)
21. Nozhov, Igor'. Prikladnoi morpfologicheskii analiz [Applied morphological analysis].In: Pravovaia Informatika *1998,* v.4. Moscow (1998)
22. Nozhov, Igor'. Grafematicheskii i morfologicheskii moduli [Graphemic and morphological modules].In: Pravovaia Informatika. 1999, v.5. Moscow (1999)
23. Paice, C.D. Method for evaluation of stemming algorithms based on error counting. Journal of the American Society for Information Science 47 (8) (1996) 632-49
24. Paice, C.D. Another stemmer. SIGIR Forum, 24 (1990) 56-61
25. Popovič, M., Willett, P. The effectiveness of stemming for natural-language access to Slovene textual data. Journal of the American Society for Information Science, 43(5) (1992) 384-390
26. Porter, M. F. An algorithm for suffix stripping. Program 14 (1980) 130-7.
27. Rissanen, J. Stochastic Complexity in Statistical Inquiry. World Scientific Publishing, Singapore, Teaneck NJ (1989)
28. Sparck Jones, K. What is the role of NLP in text retrieval? In: Strzalkowski, T (ed.) Natural Language Information Retrieval. Kluwer, Dordrecht (1999) 1-24
29. Strzalkowski, T. et al. Evaluating Natural Language Processing Techniques in Information Retrieval: A TREC Perspective. In: Strzalkowski, T.(ed.) Natural Language Information Retrieval. Kluwer, Dordrecht (1999)
30. Wexler, M., Sheridan, P., Schäble, P. Multi-language text indexing for Internet retrieval.In: Proceedings of the 5th RIAO Conference on Computer-Assisted Information Searching on the Internet (1997)
31. Xu, J., Croft, W. Corpus-Based Stemming using Co-occurrence of Word Variants. In: ACM Transactions on Information Systems, 16(1) (1995) 61-81.