The Porter stemming algorithm: then and now

Peter Willett

Author: Peter Willett is Professor and Head of the Department of Information Studies, University of Sheffield, Sheffield, UK. E-mail: p.willett@sheffield.ac.uk

Abstract

General review

Purpose: In 1980, Porter presented a simple algorithm for stemming English language words. This paper summarises the main features of the algorithm, and highlights its role not just in modern information retrieval research, but also in a range of related subject domains.

Design: Review of literature and research involving use of the Porter algorithm.

Findings: The algorithm has been widely adopted and extended so that it has become the standard approach to word conflation for information retrieval in a wide range of languages.

Value: The 1980 paper in Program by Porter describing his algorithm has been highly cited. This paper provides a context for the original paper as well as an overview of its subsequent use.

Keywords. Conflation; Information retrieval; Porter stemming algorithm; Stemming algorithm; Suffix; Word variant

Word length: 2130

1. Introduction

Natural language texts typically contain many different variants of a basic word. Morphological variants (e.g., COMPUTATIONAL, COMPUTER, COMPUTERS, COMPUTING *etc*.) are generally the most common, with other sources including valid alternative spellings, mis-spellings, and variants arising from transliteration and abbreviation. The effectiveness of searching, most obviously but not exclusively in terms of recall, would be expected to increase if it were possible to *conflate* (i.e., to bring together) the variants of a given word so that they could all be retrieved in response to a query that specified just a single variant.

In English, and many related languages, morphological variation takes place at the right-hand end of a word-form (Sproat, 1992), and this has spurred the use of user-directed right-hand truncation for online information retrieval. This is a very simple approach to conflation but one that requires considerable experience since two major types of error are possible. Over-truncation occurs when too short a stem remains after truncation and may result in totally unrelated words being conflated to the same root, as with both MEDICAL and MEDIA being retrieved by the root MED*. Under-truncation, conversely, arises if too short a string is removed and may result in related words being described by different strings, as with BIBLIOGRAPHICALLY being truncated to BIBLIOGRAPHIC, rather than to the shorter root BIBLIOGRAPH* that would also encompass BIBLIOGRAPHY.

A fully automated alternative to truncation is provided by a *stemming algorithm* (Hooper and Paice, 2005; Porter, 2001). This reduces all words with the same root to a single form, the *stem*, by stripping the root of its derivational and inflectional affixes; in most cases, only suffixes that have been added to the right-hand end of the root are removed and this approach to conflation forms the basis of the present paper. The removal of prefixes (i.e., strings that have been added at the left-hand end of a root) have been much less studied in the case of English-language retrieval; it is, however, of importance in other languages such as Malay (Ahmad *et al*., 1996).

Lovins (1968) described the first stemmer to be developed specifically for information-retrieval applications and introduced the idea of stemming based on a dictionary of common suffixes, such as *SES, *ING or *ATION. This algorithm spurred the development of many subsequent algorithms (Lennon *et al*., 1981; Porter, 2005) and, more generally, the use of stemming as a general tool in information retrieval (Frakes and Fox, 2003; Harman, 1991; Hull, 1996; Krovetz, 2000). When a word is presented for stemming in a dictionary-based stemming algorithm, the right-hand end of the word is checked for the presence of any of the suffixes in the dictionary. If a suffix is found to be present, it is removed, subject to a range of context-sensitive rules that forbid, *e.g.*, the removal of *ABLE from TABLE or of *S from GAS; in addition, a range of recoding rules may be provided to enable the conflation of variants such as FORGETTING and FORGET or ABSORB and ABSORPTION.

An alternative, very much simpler procedure was described by Porter (1980) in a study that continues to be widely cited and that has provided the inspiration for many subsequent algorithms, not just for English but also for other languages. The Porter algorithm is discussed in the remainder of this paper.

2. The Porter algorithm then

The Porter algorithm differs from Lovins-type stemmers in two major ways. The first difference is a significant reduction in the complexity of the rules associated with suffix removal. The need for simplicity is exemplified by Lovins' algorithm, which contains no less than 294 suffixes, each of which is associated with one of 29 context-sensitive rules that determine when that suffix can or cannot be removed from the end of a word; the algorithm also contains 35 recoding rules (Lovins, 1968). Despite the large number of suffixes, relatively few of them are plural forms and both the suffixes and the recoding rules suggest that the Lovins algorithm has been designed principally for the processing of scientific texts (Porter, 2005). The second difference is the use of a single, unified approach to the handling of context. Many of Lovins' context-sensitive rules relate to the length of the stem remaining after the removal of a suffix: the minimal acceptable length is normally just two characters, with a consequent risk of significant over-stemming.

There are various versions of the Porter algorithm but they differ only slightly (Porter, 2005); here, we focus on that described in the original *Program* paper (Porter, 1980). The algorithm is very simple in concept, with *ca.* 60 suffixes, two recoding rules and a single type of context-sensitive rule to determine whether a suffix should be removed. Rather than rules based on the number of characters remaining after removal, Porter uses a minimal length based on the number of consonant-vowel-consonant strings (the *measure*) remaining after removal of a suffix. This idea, which may be regarded as an easily computable representation of a syllable, was first studied by Dolby and Resnikoff (1964). A typical rule is thus as follows:

$$(m>0) \text{ *FULNESS} \rightarrow \text{*FUL}$$

This means that the suffix *FULNESS should be replaced by the suffix *FUL if, and only if, the resulting stem has a non-zero measure (*m*).

The use of only *ca*. one-fifth of the suffixes listed in Lovins' dictionary is sufficient for effective stemming since Porter's algorithm is iterative in nature, i.e., it allows a long, multi-component suffix to be removed in stages. For example, there is a rule

$$(m>0)\ \text{*FUL} \rightarrow \text{null},$$

which means that the suffix *FUL should be replaced by the null string if, and only if, the resulting stem has a non-zero measure. This rule is invoked after that involving the suffix *FULNESS given above, and thus the word HOPEFULNESS will be stemmed first to HOPEFUL and then to HOPE in the second iteration.

In all there are five steps in the algorithm: the first handles inflectional suffixes, the next three handle derivational suffixes, and there is then a final recoding step. Despite the simplicity of the basic design, early studies by both Porter (1980) and Lennon *et al*. (1981) showed that the algorithm was at least as effective as other, more complicated conflation procedures, and it was rapidly adopted by the information-retrieval research community.

3. The Porter algorithm now

Porter's algorithm was developed for the stemming of English-language texts but the increasing importance of information retrieval in the 1990s led to a proliferation of interest in the development of conflation techniques that would enhance the searching of texts written in other languages. By this time, the Porter algorithm had become the standard for stemming English, and it hence provided a natural model for the processing of other languages. In some of these new algorithms the only relationship to the original is the use of a very restricted suffix dictionary (Porter, 2005), but Porter himself has developed a whole series of stemmers that draw on his original algorithm and that cover Romance (French, Italian, Portuguese and Spanish), Germanic (Dutch and German) and Scandinavian languages (Danish, Norwegian and Swedish), as well as Finnish and Russian (Porter, 2006).

These stemmers are described in a high-level computer programming language, called Snowball (Porter, 2006) that has been developed to provide a concise but unambiguous description of the rules for a stemmer. Some non-English stemmers can operate effectively using simple sets of rules, with Latin being perhaps the best example of a language that is defined in what is essentially algorithmic form (Schinke *et al.*, 1996). However, this level of regularity and simplicity is by no means common; in such cases, Snowball provides a concise but powerful description that can then be processed by a compiler to give a C or Java implementation of the algorithm for the chosen language (Porter, 2001). In passing, it is worth noting that this paper by Porter contains an extremely illuminating discussion of stemming and the structures of words that is very well worth reading, even if one does not wish to obtain any of the downloadable programs.

These developments of the Porter algorithm can only serve further to increase the level of knowledge and understanding of the original, English-language version; this level is already considerable as is evidenced by the following simple citation analysis. While the precise relationship between citation and significance is a matter of some dispute, it does seem reasonable to regard the 1980 *Program* paper as being a significant contribution to the literature since a search of the *ISI Web of Knowledge* database on 21st March 2006 yielded 442 citations. Hardly surprisingly, many of these appeared in mainstream information science journals (e.g., *Journal of Documentation*, *Information Processing and Management*, *Information Retrieval*, the *Journal of the American Society for Information Science and Technology*, and *Scientometrics*); however, the majority were in the more general computer science literature relating to data and knowledge (e.g., *Artificial Intelligence Review*, *IEEE Transactions on Knowledge and Data Engineering*, *International Journal of Intelligent Systems*, *Lecture Notes in Computer Science*, and *Pattern Recognition Letters*), with some coming from still more widely dispersed fields (e.g., *Behaviour Research Methods*, *Bioinformatics*, *Neuroinformatics*, *Sociological Methodology* and *User-Modeling and User-Adapted Interaction*). It is interesting to note that almost 100 of the citations appeared in 2005 or 2006 (with all of the journals noted above carrying citations in this period), from which we can conclude that the paper continues to be of importance, despite it first being published over a quarter of a century ago.

4. Conclusions

Porter's algorithm is important for two reasons. First, it provides a simple approach to conflation that seems to work well in practice and that is applicable to a range of languages. Second, it has spurred interest in stemming as a topic for research in its own right, rather than merely as a low-level component of an information retrieval system. The algorithm was first published in 1980; however, it and its descendants continue to be employed in a range of applications that stretch far beyond its original intended use.

References (All URLs were checked 18th April 2006)

Ahmad, F., Yusoff, M. and Sembok, T.M.T. (1996), "Experiments with a stemming algorithm for Malay words", *Journal of the American Society for Information Science*, Vol. 47 No. 12, pp. 909-918.

Dolby, J.L. and Resnikoff, H.L. (1964), "On the structure of written English", *Language*, Vol. 40 No.2, pp. 167-196.

Frakes, W.B. and Fox, C.J. (2003), "Strength and similarity of affix removal stemming algorithms", *SIGIR Forum*, Vol. 37, pp. 26-30. Available at: http://www.sigir.org/forum/S2003/StemSim.pdf.

Harman, D. (1991), "How effective is suffixing?", *Journal of the American Society for Information Science*, Vol. 42 No 1, pp. 7-15.

Hooper, R. and Paice, C. (2005), *The Lancaster Stemming Algorithm*. Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/

Hull, D.A. (1996), "Stemming algorithms: a case study for detailed evaluation", *Journal of the American Society for Information Science*, Vol. 47 No. 1, pp. 70-84.

Krovetz, B. (2000), "Viewing morphology as an inference process", *Artificial Intelligence*, Vol. 118 Nos. 1 and 2, pp. 277-294.

Lennon, M., Peirce, D.S., Tarry, B.D. and Willett, P. (1981), "An evaluation of some conflation algorithms for information retrieval", *Journal of Information Science*, Vol. 3 No.4, pp. 177-183.

Lovins, J.B. (1968), "Development of a stemming algorithm", *Mechanical Translation and Computational Linguistics*, Vol. 11 Nos 1 and 2, pp. 22-31.

Porter, M.F. (1980), "An algorithm for suffix stripping", *Program*, Vol. 14 No.3, pp. 130-137.

Porter, M.F. (2001), *Snowball: A Language for Stemming Algorithms*. Available at: **http://www.snowball.tartarus.org/texts/introduction.html**.

Porter, M.F. (2005), "Lovins revisited", In Tait, J.I. (editor) *Charting a New Course: Natural Language Processing and Information Retrieval. Essays in Honour of Karen Spärck Jones*, Springer, Dordrecht, pp. 39-68.

Porter, M.F. (2006) "Stemming algorithms for various European languages". Available at: **http://www.snowball.tartarus.org/texts/stemmersoverview.html**

Schinke, R., Greengrass, M., Robertson, A.M. and Willett, P. (1996), “A stemming algorithm for Latin text databases”, *Journal of Documentation*, Vol. 52 No.2, pp. 172-187.

Sproat, R. (1992), *Morphology and Computation*, MIT Press, Cambridge MA.