Stemming Algorithms for Information Retrieval and Question/Answer Systems

Stemming refers to the process of removing affixes (prefixes and suffixes) from words. In the information retrieval context, stemming is used to conflate word forms to avoid mismatches that may undermine recall. As a simple example, consider searching for a document entitled "How to write". If the user issues the query "writing" there will be no match with the title. However, if the query is stemmed, so that "writing" becomes "write", then retrieval will be successful. In many languages stemming is imperative for retrieval performance. For instance, in Hebrew, stemming increases the number of documents retrieved by between 10 and 50 times (Krovetz 1993). In English gains tend to less dramatic. Nonetheless, stemming has been shown to produce reliable retrieval improvement (Krovetz 1993, Hull 1996, c.f. Harman 1991).

Furthermore, affixes often carry information such as part of speech, plurality, and/or tense that is crucial for the development of more sophisticated question/answer information systems. For instance, consider the sentence "The stugy nitfels untrooled the drutable jupan." From the word order and affixes, alone, you know that more than one thing did something to one thing. You know this happened in the past and that what was done was the opposite of trooling. Question/answer systems will rely on such structural cues and hence will require a high precision stemmer as a preprocessing step.

The most widely cited stemming algorithm was introduced by Porter (1980). The Porter stemmer applies a set of rules to iteratively remove suffixes from a word until none of the rules apply. The Porter stemmer has a number of well-documented limitations (Krovetz 1993, Xu & Croft 1998, Arampatzis, van der Weide, Bommel & Koster 1999). First, it is not restricted to produce word stems. So, for instance, "general" becomes "gener" and "iteration" becomes "iter". As a consequence, the stemmer can conflate words that have very different meanings (or senses). Second, even when it does produce a word stem it is often overzealous. "Doing" becomes "doe" and "punish" becomes "pun". Finally, like many existing stemmers it ignores prefixes completely, so "reliability" and "unreliability" remain as unrelated tokens. The Lovins stemmer (Lovins 1968) is similar in mechanism but has a larger set of suffixes (each of which may included multiple morphemes) and does not apply its rules iteratively. While it tends to be more conservative than the Porter stemmer still suffers from over conflation and non-word stems.

Krovetz (1993) argued that meaning is essential to the stemming decision. He attempted to resolve some of the limitations of the Porter and Lovins stemmers, by restructuring the rule set so that it would produce word stems, using a non-iterative stripping mechanism and by checking a dictionary for the current string before removing a suffix. Typically, a dictionary will list word forms separately only if they have different meanings. However, the Krovetz algorithm relies heavily on the integrity of the dictionary and tends to be too conservative. So, "predictions" becomes "prediction", while "prediction" becomes "predict" and they are not conflated when they should be.

Xu and Croft (1998) used a normalized co-occurrence mechanism to make inferences about whether word forms were of the same meaning. They created a corpus-specific stemmer that showed improved retrieval performance and was able to capture some subtle meaning related effects. For instance, their stemmer did not conflate the words "gas" and "gases". In the Wall Street Journal corpus, on which they were working, "gases" was used in the sense of "inert gases" or "hot gases", while "gas" nearly always referred to petrol. For this corpus, then, conflation was undesirable.

The Xu and Croft (1998) algorithm does not actually remove suffixes, but instead defines equivalence classes of words that should be conflated. Finding these equivalence classes can be computationally expensive and does not identify the stem and suffixes independently making it unusable in the question/answer context outlined above.

At the Key Centre for Human Factors and Applied Cognitive Psychology we are currently working on a stemming algorithm that leverages algorithms for creating meaning representations (Latent Semantic Analysis, LSA, Landauer, Foltz, & Laham 1998 and Hyperspace Analog to Language, HAL, Lund & Burgess 1996) to produce a method that is computationally efficient and suitable as a preprocessing step in question/answer systems. The algorithm uses a set of prefix and suffix stripping rules, but applies them in parallel generating multiple possible "parses" of the word. For instance, the word "prediction" would be parsed both as "pre diction" and "predict ion". As a consequence, we are tentatively calling the algorithm the parallel or P stemmer. The preferred parse is selected using a number of heuristics including:

1. choose longest stem
2. choose highest frequency stem
3. choose stem that has highest normalized co-occurrence with original word using the em measure (Xu & Croft 1998)
4. choose stem that has the closest meaning vector (using LSA or HAL) to the original word

The current project involves assessing the P stemmer (including the various heuristic possibilities) against existing stemmers including the Porter stemmer and the Krovetz stemmer. In the past, stemmer assessment has always been done in terms of the impact on retrieval performance rather than on the actual number of correct stemmings. As we are interested in applying the algorithm in non-traditional information retrieval circumstances, such as question/answer systems, we intend to measure the performance of the stemmers in terms of the number of correct stemmings produced.

References

Arampatzis, A, van der Weide, Th.P., Koster, C.H.A., and van Bommel, P. (2000). Linguistically-motivated Information Retrieval. Encyclopedia of Library and Information Science, published by Marcel Dekker, Inc. - New York – Basel. To appear.

Harman, D. (1991). How effective is suffixing? Journal of the American Society for Information Science, 42(7), 7-15.

Hull, D. (1996). Stemming algorithms: A case study for detailed evaluation. Journal of the American Society for Information Science, 47(1), 70-84.

Krovetz, R. (1993). Viewing morphology as an inference process. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 191-202.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes, 25, 259-284.

Lovins, J. B. (1968). "Development of a Stemming Algorithm", Mechanical Translation and Computational Linguistics, 11.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instrumentation, and Computers, *28*, 203-208.

Porter, M. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.

Xu, J. and Croft, B. (1998). Corpus-Based Stemming using Co-occurrence of Word Variants. ACM Transactions on Information Systems, 16 (1).