**An Algorithm for suffix stripping**

**Problem of statement:**

In the realm of natural language processing and information retrieval, the main goal is to make text analysis more efficient and accurate. That's why stripping words down to their basic form by cutting off endings is crucial for information retrieval. This paper explains the core concept of suffix stripping.

**Assumption in prior work:**

Words that share a common root often mean something alike. When we take away the endings of words, it's important to consider what's left of the word. For example, we only remove the ending if the word that remains is likely to be a proper root in the language. Sometimes, changing the ending of a word changes how it's spelled. In any program that removes word endings for information retrieval, the goal is to enhance information retrieval, not simply to practice linguistics. However, relying solely on lists of endings with different rules might not always be very effective and might have a success rate lower than 1.

**Idea:**

The optimal approach to remove suffixes from multiple words and merge them into a coherent stream is a key focus. Additionally, the author concentrates on words whose meanings evolve with suffix additions. Words are comprised of vowels and consonants, making the grouping of these elements pivotal for accurate stripping. The author presents a mathematical equation:

Word = Cm [V], with 'm' denoting the word's measure.

**Techniques:**

To outline the suffix stripping algorithm, the author employs the following strategies:

Determine the value of 'm,' representing the number of VC (vowel-consonant) sequences in a word. For instance, in the word IVY, m equals 1.

Implement rules in a systematic sequence:

Step 1a: Address plurals and past participles (e.g., caresses -> caress).

Step 1b: Manage words ending in 'ed' and 'ing' (e.g., hopping -> hop).

Step 1c: Handle words ending in 'y' (e.g., happy -> happi).

Step 2: Address various double suffixes (e.g., ational -> ate).

Step 3: Remove specific suffixes (e.g., icate -> ic).

Step 4: Remove particular suffixes if specific conditions are met (e.g., when 'm' is greater than 1), such as 'al,' 'ance,' and 'ence.'

Step 5: Make final adjustments to eliminate a trailing 'e' and manage 'll' (e.g., probate -> probat).

**Evaluation:**

The algorithm's performance has undergone thorough evaluation concerning efficiency, accuracy, and its influence on information retrieval tasks. However, it's worth noting potential instances of over- or under-stripping, as seen with words like UNIVERSAL and UNIVERSE. Additionally, language limitations and the introduction of new syntax could potentially impact the algorithm's effectiveness.

**Implication:**

The Porter Stemming Algorithm stands as a fundamental tool in English text natural language processing. Its notable efficiency and effectiveness guarantee its ongoing significance upon implementation. This solidifies its role as a fundamental pillar in the realm of suffix stripping. Overall, the algorithm is lauded for its effectiveness, efficiency, speed, accuracy, and simplicity, particularly in the context of Information Retrieval (IR).

**Review by:**

**Arun Singh Dhanadi**