Stringified Numbers to Words Conversion Implementation Discussion

Introduction:

This implementation discussion outlines the approach, time complexity of the solution implemented and other possible solutions to solve this issue and why I chose to implement with this method.

Algorithm of choice:

1. **Input Handling:**

- The input number is received as a string, which may include both integer and decimal parts. The number may also be in negative.

2. **Integer Part Conversion:**

- The integer part of the input number is processed.
- It is divided into groups of three digits, starting from the least significant digits.
- Each group is converted into words separately.

3. **Conversion of Each Group:**

- For each group of three digits:
    - The hundreds, tens, and units places are extracted.
    - Corresponding words for each digit are retrieved from predefined arrays.
    - Special cases for teens and tens are handled.
    - Magnitude terms (thousands, millions, etc.) are appended based on the position of the group.
    - The words for the digits and magnitude terms are combined to form the final word representation of the group.

4. **Decimal Part Conversion:**

- If present, the decimal part of the input number is processed similarly to the integer part.
- Words for negative, dollar(s) and cent(s) are added to the final result.

5. **Combining Results:**

- The word representations of the integer and decimal parts (if applicable) are combined to form the complete word representation of the input number.

6. **Output:**

- The final word representation of the input number, including both integer and decimal parts (if applicable), is returned as a string.

Time complexity:

O(N).

Other possible solution:

Upon research on this problem, there are a few other ways of tackling this issue without the use of libraries or packages. One of it is through **recursion**. Although it may offer flexibility and scalability, it will also introduce complexity in the logic which may be harder to understand for some developers. Recursion may also require more memory and processing power compared to linear approaches. Hence, which is why I prefer to implement my linear solution instead as it is easier to understand and maintain.