```
Help on module solver: Text output generated using pydoc3.8, reformatted into a PDF document.

NAME
   solver

DESCRIPTION
    AUTHOR: James Akl
    CONTACT: james-akl@outlook.com

FUNCTIONS
    generate_lookup(wordlist_path: str) → None
        Write to disk the lookup table (associated with the specified word list) containing two dictionaries: anagrams and vectors.

        The input file (the word list) is read from disk and the output file (the lookup dictionaries) is written to disk.
        The generated anagrams dictionary maps a sorted key to its anagrams (e.g., "acr" to ["arc", "car"]).
        The generated vectors dictionary maps a sorted key to its letter count (e.g., "acr" to [1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]).
        Assumption: When reading the word list's text file, assume one word token per line.

    get_results(search_input: str) → Set[str]
        Return the set containing all valid subanagrams for the search input string.

        The procedure is outlined as follows:
            1. Load from disk the anagrams and vectors dictionaries.
            2. Compare the vector of each candidate key with that of the search_input (here, vector means letter count).
            3. Subanagram test: A passing candidate key must have a count less or equal to the search_input for each of 26 letters.
            4. Upon passing, include (in the solution set) all anagrams associated with that passing candidate key.
            5. After performing this for all candidate keys, remove (from the solution set) the word identical to search_input.
            6. Return the solution set.

    main() → None
        Solve the subanagram search problem for the specified input and print to the CLI the results ordered by word length.

        This is performed over four steps:
            1. Parse and store the CLI arguments.
            2. If unavailable, generate the lookup table, using the specified word list.
            3. Obtain the solutions for the input, using the lookup table.
            4. Print the solution subanagrams, ordered by word length in decreasing order (longest to shortest).
    parse_args() -> argparse.Namespace
        Return the user-specified CLI arguments in an object of type argparse.Namespace.

        This helper function wraps the usage of the argparse module.
        It creates a parser object of type ArgumentParser, defines the CLI arguments with their details, and returns them.

    print_results(search_input: str, results_unordered: Set[str]) -> None
        Print to the CLI the solution subanagrams ordered by word length in decreasing order (from the longest to the shortest).

    vectorize_word(word: str) -> List[int]
        Return a character-count list representation of the input word.

        The outputs are vectors in the 26-dimensional vector space whose basis is the lowercase Latin alphabet.
        Example: both "abc" and "cab" map to [1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0].

DATA
    Dict = typing.Dict
    List = typing.List
    Set = typing.Set
    ascii_lowercase = 'abcdefghijklmnopqrstuvwxyz'

FILE
   solver.py
```