Summary

MeCab: widely used for Japanese tokenization. Algorithm for analysis: Bigram Malkov model. Algorithm for the dictionary: double array. Using Conditional Random Fields (CRFs) to estimate parameters. Library in Python and C++.
Focusing on Japanese only and using the dictionary and pre-trained model for Japanese. Fast and efficient.
https://github.com/SamuraiT/mecab-python3
license:`mecab-python3` is copyrighted free software by Taku Kudo taku@chasen.org and Nippon Telegraph and Telephone Corporation, and is distributed under a 3-clause BSD license (see the file `BSD`). Alternatively, it may be redistributed under the terms of the GNU General Public License, version 2 (see the file `GPL`) or the GNU Lesser General Public License, version 2.1 (see the file `LGPL`).

Juman: There is Juman++ too, which uses Recurrent Neural Network Language Model (RNNLM). It has better functionality than Juman. Algorithm for analysis: bigram Malkov model. Algorithm for the dictionary: PATRICIA tree.
https://github.com/ku-nlp/juman
License: Apache license 2.0

SentencePiece: SentencePiece, on the other hand, is a tool for subword segmentation that uses a different approach. The authors describe SentencePiece as a language-independent subword tokenizer and detokenizer designed for Neuralbased text processing [7]. While using a different approach with focus on providing a method to support language-independent multilingual text processing, its performance is shown to be effective for various tasks, such as English-Japanese neural machine translation [7].

[7] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Brussels, 2018.

The above is from the NLP meeting in 2021.
https://www.anlp.jp/proceedings/annual_meeting/2021/pdf_dir/D3-1.pdf

Tokenise.py: transducer itself to tokenize. Only tokenization, not labeling. Explanation is on github. After the initialization of the trie, it uses a finite state transducer to tokenise the text. The finite state transducer goes through the text and advances states when the state can continue on the next character, otherwise terminates the state. When the state reaches a final state, it is added to the list of marks. Invalid characters are skipped. When all states are terminated, the program does a flush. Each character in the sequence of characters parsed is added to `marks` as an invalid character to fill up any blanks. Then, the program passes the marks to a sequence generator that generates all possible sequences.

The sequence generator works from the start to the end as a recursive function. It will take in a token as part of a sequence if: 1) it is valid OR 2) it is invalid but there are no valid tokens starting with that invalid character. Once it obtains the list of sequences, it returns that to the tokeniser.

https://github.com/cl4r3/tokenisation

Japanese morphological analysis is based on the construction of lattices based on lexicographic search.
However, with the advent of general-purpose language models, methods that solve token classification problems by assigning labels to each token in a word (subword) sequence have achieved practical accuracy.
From https://www.anlp.jp/proceedings/annual_meeting/2023/pdf_dir/C2-3.pdf