```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author: Josimar H. Lopes,
# Master of EEIE, NUT


import os
import sys
from subprocess import Popen, PIPE
import tempfile
import nltk


class StanfordParserPT():
    """Class to interface with the Stanford Parser, for compatibility with Portuguese"""

    def __init__(self, encoding=None, verbose=False):
        if len(sys.argv) <= 1:
            self.fileName = raw_input("Provide an input file:: ")
        else:
            self.fileName = sys.argv[1]

        path_to_jar = "stanford-parser-2010-11-30/stanford-parser.jar"
        path_to_model = "stanford-parser-2010-11-30/cintil.ser.gz"

        if not os.path.isfile(path_to_jar):
            raise IOError("Could not locate:: file %s not found" % path_to_jar)

        if not os.path.isfile(path_to_model):
            raise IOError("Stanford Parser model file not found: %s" % path_to_model)

        self._stanfordparser_jar = path_to_jar
        self._stanfordparser_model = path_to_model
        self._encoding = encoding
        self._verbose = verbose

    def tokenize(self):

        fsents = open(self.fileName)
        nlines = sum(1 for line in open(self.fileName))
        #  print nlines
        sentWords = []

        for i in (1, nlines):
            pTok = Popen(["Tokenizer/run-Tokenizer.sh"], stdin=PIPE, stdout=PIPE)
            tokSents, stderr = pTok.communicate(fsents.readline())
            sentWords.append(tokSents.replace("*/", "").replace("\*", "").replace("\n", "").strip())
        print "i :: \n", i, sentWords

        fsents.close()
        return self.parse(sentWords)

    def parse(self, tokens):
        return self.batch_parse([tokens])  # [1]

    def batch_parse(self, sentences):

        encoding = self._encoding
        java_options='-Xmx500m'

        # Create a temporary input file
        _input_fh, _input_file_path = tempfile.mkstemp(text=True)


        # Build the java command to run the parser
        _stanfordparser_cmd = ['java',
                               java_options,
                               '-cp',
                               self._stanfordparser_jar,
            'edu.stanford.nlp.parser.lexparser.LexicalizedParser',
            '-tokenized',
            '-sentences','newline',
            '-outputFormat', 'oneline',
            '-uwModel', 'edu.stanford.nlp.parser.lexparser.BaseUnknownWordModel',
            self._stanfordparser_model,
            _input_file_path
            ]

        # Write the actual sentences to the temporary input file
        _input_fh = os.fdopen(_input_fh, 'w')
        _input = ''.join('\n'.join(x) for x in sentences)  # + '\n'

        if isinstance(_input, unicode) and encoding:
            _input = _input.encode(encoding)

        print "\ninput :: \n", _input
        _input_fh.write(_input)
        _input_fh.close()

        # Run the parser and get the output
        process = Popen(_stanfordparser_cmd, stdout=PIPE, stderr=PIPE)
        stanfordparser_output, stderr = process.communicate()

        if self._verbose:
            verbose = stderr.decode(encoding).strip().split("\n")
            print "%s\n%s" % (verbose[0], verbose[-1])
        if encoding:
            stanfordparser_output = stanfordparser_output.decode(encoding)

        print "\nstanford_output ::\n", stanfordparser_output

        # Delete the temporary file
        os.unlink(_input_file_path)

        # Output the parse trees
        parse_trees = []
        for string_tree in stanfordparser_output.strip().split("\n"):
            print "string :: ", string_tree
            parse_trees.append(nltk.Tree.fromstring(string_tree.replace("(ROOT", "")[:-1]))
        return parse_trees

if __name__ == '__main__':

    parser = StanfordParserPT(encoding="utf-8")
    tree = parser.tokenize()
    for t in tree:
        print t, '\n'
        t.draw()
```