LAMPIRAN

1. *Script* Normalisasi Teks dan Implementasi Jaro-Winkler (JaroWinkler.py)

```
import nltk
import re
import string
import time
import multiprocessing
from pyjarowinkler import distance

nltk.data.path.append('/home/grelly/venvs/skripsi/')

class SpellCheck:
    def __init__(self):
        self.correct_words = []

    # word tokenize
    def tokenize_text(self, text):
        tokens     =     [token.strip()     for     token     in
nltk.word_tokenize(text)]
        return tokens

    # function for normalize document
    def normalize_document(self, text):
        text = self.tokenize_text(text)
        return text

    # dataset loader
    def read_dictionary(self):
        dictionary = set()
        if dictionary:
            return

        with open('/home/grelly/skripsi/dataset4') as file:
            dataset = file.readlines()
        kbbi = set([data.strip() for data in dataset])
        return kbbi

    # check correct word in document
    def check_correct_word(self, document, output):
        winkler = True
        scaling = 0.1
        correct_word = None

        wordlist = self.read_dictionary()
        for word in wordlist:
            pylibjaro     =     distance.get_jaro_distance(word,
document, winkler, scaling)

            if pylibjaro == 1.0:
                correct_word = document
                break
        output.put(correct_word)
```

```
    # check status of each word based on existing correct words
    def check_all_words(self, word, output, i):
        if word not in self.correct_words:
            check_word = {"id": i + 1, "text": word, "status":
False}
        else:
            check_word = {"id": i + 1, "text": word, "status":
True}
        output.put(check_word)

    # run process in parallel
    def jaro_parallel_runner(self, inputs):
        output = multiprocessing.Queue()
        start_time = time.time()
        results = {}
        all_words = []

        # multiprocessing for checking correct words
        processes_correct                                        =
[multiprocessing.Process(target=self.check_correct_word,
args=(inputs[i], output))
                              for i in range(len(inputs))]

        for process in processes_correct:
            process.start()

        for process in processes_correct:
            process.join()

        result     =     [output.get()     for     process     in
processes_correct]
        self.correct_words = list(filter(None, result))

        # multiprocessing for checking all words status based
on existing correct words
        for i in range(len(inputs)):
            process                                              =
multiprocessing.Process(target=self.check_all_words,
args=(inputs[i], output, i))
            process.start()
            process.join
            result2 = output.get()
            all_words.append(result2)

        results['result'] = all_words
        results['number_of_words'] = len(inputs)
        results['correct'] = len(self.correct_words)
        results['typo']          =          (len(inputs)          -
len(self.correct_words))
        results['time']  =  '{}  seconds'.format(time.time()  -
start_time)

        return results
```

2. Script Implementasi Model pada *Web Service* (Main.py)

```
from flask import Flask, jsonify, request
from flasgger import Swagger
from sklearn.externals import joblib
from flask_cors import CORS

app = Flask(__name__)
Swagger(app)
CORS(app)

@app.route('/input/check-words', methods=['POST'])
def checkWords():
    """
    Check sll the words from document here
    ---
    tags:
        - Rest Controller
    parameters:
        - name: body
          in: body
          required: true
          schema:
            id: Words
            required:
                - text
            properties:
                text:
                    type: string
                    description: input text here
    responses:
        200:
            description: input success
    """
    inputs = request.get_json()

    data = inputs['text']

    model    =    joblib.load('/home/grelly/skripsi/jaro-winkler-
spellcheck.pkl')
    normalize_data = model[0].normalize_document(data)
    result = model[0].jaro_parallel_runner(normalize_data)

    return  jsonify({'results':  result,  'message':  'process
success'})

@app.route('/input/parse-to-text', methods=['POST'])
def parseToText():
    """
    Check sll the words from document here
    ---
    tags:
        - Rest Controller
    parameters:
        - name: body
          in: body
```

```
            required: true
            schema:
              id: Words
              required:
                  - text
              properties:
                  text:
                        type: string
                        description: input text here
    responses:
        200:
            description: input success
    """

    input = request.get_json()

    file = input['text']

    pkl_load      =      joblib.load('/home/grelly/skripsi/parse-
file.pkl')
    result = pkl_load[0].readPDFDocument(file)

    return jsonify({'results': result})

if __name__ == '__main__' :
    app.run(debug=True)
```