Pengenalan dan Klasifikasi Tulisan pada Nota Pembelian Material (Studi Kasus Proyek Konstruksi)

RM Ivan Indrakusuma, Adhatus Solichah Ahmadiyah, dan Nurul Fajrin Ariyani
Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: adhatus@if.its.ac.id

***Abstrak*—CV Bangun Karya Sejahtera merupakan perusahaan yang aktif bergerak di bidang properti dan memiliki banyak arus kas masuk maupun keluar. Perusahaan telah memiliki sistem informasi untuk membantu memantau aliran dana. Namun, karyawan harus memasukkan data tersebut secara manual ke dalam sistem. Proses input ini seringkali memakan banyak waktu mereka. Oleh karena itu, tugas akhir ini bermaksud untuk menambahkan fitur pada sistem informasi yang ada untuk membantu karyawan dalam menginput data secara otomatis ke dalam sistem informasi perusahaan yang ada. Fitur ini memanfaatkan teknologi Optical Character Recognition (OCR). Ini dikembangkan menggunakan kerangka Flask berdasarkan bahasa pemrograman Python. Klasifikasi dilakukan dengan menggunakan Support Vector Machine. Hasil penelitian menunjukkan bahwa fitur tambahan dapat melakukan pengenalan dan klasifikasi teks dengan SGDClassifier rata-rata 8,89 detik. Adanya fitur OCR pada sistem informasi ini diharapkan dapat membantu karyawan perusahaan dalam menginput data ke dalam sistem informasi CV Bangun Karya Sejahtera.**

***Kata Kunci— Flask, OCR, Sistem Informasi, Web*.**

I. PENDAHULUAN

SERING kali pekerja kantoran menghabiskan banyak waktu dalam melakukan pekerjaan mereka, tetapi hal tersebut berbanding terbalik dengan hasil dari kerja mereka. Hal tersebut terjadi karena banyak waktu pekerja dihabiskan untuk memasukkan data ke dalam program untuk selanjutnya diolah. Terlebih lagi hal yang umum dilakukan untuk memasukkan data tersebut adalah dengan mengetik secara manual.

Kemajuan teknologi informasi yang sudah sangat pesat memiliki manfaat yang besar bagi kehidupan manusia saat ini. Terutama teknologi yang dapat dimanfaatkan dalam membaca data fisik ke dalam data yang bisa dibaca oleh komputer. Teknologi yang dimaksudkan adalah *Optical Character Recognition* (OCR). Berdasarkan hal tersebut, penulis tertarik untuk mengembangkan *existing system* yang pernah penulis bangun saat melakukan Kerja Praktik (KP), yaitu Sistem Informasi Monitoring Proyek Konstruksi. Teknologi ini dapat digunakan dalam sistem informasi tersebut untuk mengekstrak data dari foto. Data yang diekstrak oleh *Optical Character Recognition* (OCR) kemudian akan diklasifikasikan ke dalam *input field* yang sesuai.

Hasil yang diharapkan nantinya adalah pengembangan Sistem Informasi Monitoring Proyek yang akan diimplementasikan dengan menggunakan teknologi *Optical Character Recognition* (OCR) untuk mengekstraksi data foto nota. Kemudian data yang sudah diekstraksi tersebut akan diklasifikasikan sesuai dengan *input field* yang ada.

II. TINJAUAN PUSTAKA

A. Sistem Informasi Proyek Konstruksi

Sistem Informasi Monitoring Proyek Konstruksi adalah sistem informasi yang penulis kembangkan pada saat Kerja Praktik. Sistem Informasi tersebut berfungsi untuk membantu karyawan perusahaan dalam melakukan manajemen keuangan, dalam hal ini, neraca keuangan dan rencana anggaran belanja perusahaan.

Perbedaan sistem yang akan dibangun dengan existing sistem terdapat pada proses input, dimana pada sistem lama hanya menggunakan input field atau form untuk masukan datanya, sehingga proses input data dilakukan secara manual. Sedangkan untuk sistem baru akan digunakan upload nota belanja, sehingga data yang dibutuhkan akan langsung diambil dari foto nota belanja.

B. Optical Character Recognition

Optical Character Recognition (OCR) adalah teknologi yang digunakan untuk mengenali teks di dalam gambar, seperti dokumen dan foto. Teknologi OCR digunakan untuk mengubah hampir semua jenis gambar yang mengandung teks tertulis menjadi data teks yang dapat dibaca mesin. Implementasi *Optical Character Recognition* (OCR) dilakukan pada proses pembacaan teks dari foto nota. OCR pada sistem ini memanfaatkan salah satu *library* yang umum digunakan, yaitu Pytesseract. Pembahasan mengenai Pytesseract akan dijelaskan pada subbab selanjutnya.

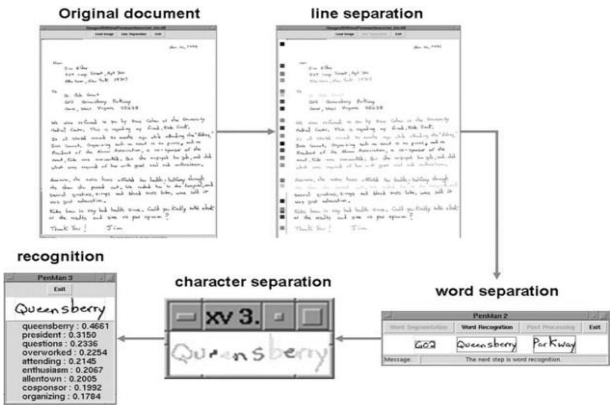
C. Pytesseract

Pytesseract atau *Python-tesseract* adalah alat *Optical Character Recognition (OCR)* untuk Python. Artinya, pytesseract akan mengenali dan membaca teks yang disematkan dalam gambar. Python-tesseract adalah pembungkus untuk mesin Tesseract-OCR Google. Pytesseract juga berguna sebagai skrip panggilan yang berdiri sendiri untuk Tesseract, karena dapat membaca semua jenis gambar yang didukung oleh *library* pencitraan, termasuk jpeg, png, gif, bmp, tiff, dan lainnya. Pytesseract ini memiliki kelebihan pada pembacaan teks cetak.

Pytesseract pada sistem ini digunakan pada saat proses rekognisi tulisan dari foto. Fungsi yang akan dipakai dalam Tugas Akhir ini adalah fungsi *image_to_string* yang berfungsi untuk mengekstrak data pada gambar ke dalam bentuk *string*.

D. Scikit-learn

Scikit-learn adalah modul Python yang mengintegrasikan berbagai algoritma machine learning canggih untuk problem



Gambar 1. Proses pengenalan tulisan tangan.

supervised dan unsupervised skala menengah. Paket ini berfokus pada penggunaan machine learning oleh non-spesialis dengan menggunakan level bahasa tingkat tinggi. Penekanan diletakkan pada kemudahan penggunaan, kinerja, dokumentasi, dan konsistensi API. Library ini memiliki dependency minimal dan didistribusikan di bawah lisensi BSD. Hal tersebut mendorong penggunaannya dalam bidang akademik dan komersial [1]. Pada sistem ini, modul scikit-learn digunakan dalam melakukan proses klasifikasi.

E. Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah model *machine learning supervised* yang menggunakan algoritma klasifikasi untuk masalah klasifikasi dua grup. Setelah memberikan set model SVM dari data pelatihan berlabel untuk setiap kategori, SVM dapat mengkategorikan teks baru.

Dibandingkan dengan algoritma yang lebih baru seperti *Neural Network*, SVM memiliki dua keuntungan utama: kecepatan yang lebih tinggi dan kinerja yang lebih baik dengan jumlah sampel yang terbatas (ribuan). Ini membuat algoritma ini sangat cocok untuk masalah klasifikasi teks, di mana biasanya memiliki akses ke kumpulan data beberapa ribu sampel yang diberi *tag* atau label. Implementasi *Support Vector Machine* pada sistem ini adalah pada saat dilakukannya klasifikasi data.

F. Flask

Flask adalah kerangka kerja aplikasi web WSGI yang ringan. *Flask* dirancang untuk memulai pembuatan web dengan cepat dan mudah, dan mempunyai kemampuan dalam pembuatan aplikasi yang lebih kompleks. Flask dimulai sebagai wrapper sederhana di sekitar Werkzeug dan Jinja dan telah menjadi salah satu kerangka kerja aplikasi web Python paling populer.

G. Pengenalan Tulisan Tangan

Pengenalan tulisan tangan dilakukan setelah penulisan selesai dengan mengubah dokumen tulisan tangan menjadi bentuk digital. OCR memainkan peran penting untuk perpustakaan digital, memungkinkan masuknya informasi tekstual gambar ke komputer dengan metode digitalisasi, restorasi gambar, dan pengenalan. Gambar 1 menunjukkan proses pengenalan tulisan tangan [2].

Sistem pengenalan tulisan tangan umumnya terdiri dari empat proses: akuisisi, segmentasi, pengenalan, dan pasca-pemrosesan (Gambar 1). Pertama, tulisan tangan yang akan dikenali didigitalkan melalui pemindai atau kamera. Kedua, citra dokumen tersegmentasi menjadi garis, kata, dan karakter individu. Ketiga, setiap karakter dikenali menggunakan teknik OCR. Akhirnya, kesalahan dikoreksi menggunakan leksikon atau pemeriksa ejaan.

Tabel 1.
Perbedaan dengan aplikasi existing

No.	Proses	Aplikasi Sebelumnya	Aplikasi akan Dikembangkan
1	OCR untuk membaca nota dan ekstrak data	Tidak tersedia	Tersedia
2	Pengisian form pengeluaran	Manual	Terdapat pilihan manual dan otomatis
3	Kerangka Kerja	Laravel	Flask
4	Navigasi	Menggunakan Navbar di atas	Menggunakan Sidebar

Sedangkan pada sistem pengenalan teks menerima masukan berupa gambar yang berisi beberapa informasi teks. Keluaran dalam format elektronik yaitu informasi teks dalam gambar adalah disimpan dalam bentuk yang dapat dibaca komputer. Sistem pengenalan teks dapat dibagi dalam modul berikut: (1) pra-pemrosesan, (2) pengenalan teks, (3) pasca-pemrosesan.

Pada modul Pra-pemrosesan, dokumen kertas umumnya dipindai oleh optic pemindai dan diubah menjadi bentuk gambar. Pada tahap ini gambar dianalisis lebih lanjut. Jadi untuk meningkatkan kualitas gambar input, beberapa operasi dilakukan untuk peningkatan citra seperti penghilangan noise, normalisasi, binarisasi, dll. Pada modul Pengenalan Teks digunakan untuk pengenalan teks. Oleh karena itu dalam modul ini teknik segmentasi, ekstraksi fitur, dan kemudian yang terakhir adalah klasifikasi. Pada modul Pasca Pemrosesan, keluaran dari modul sebelumnya adalah berupa data teks yang dimengerti oleh komputer, sehingga perlu untuk menyimpannya ke dalam beberapa format yang tepat (seperti MS-Word) untuk penggunaan selanjutnya [3].

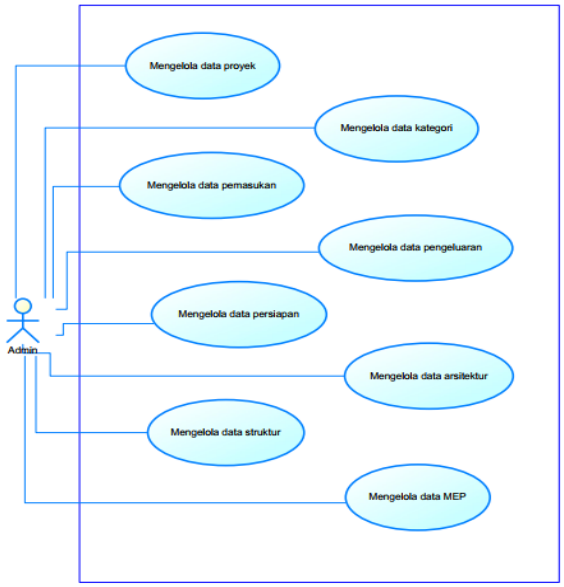
H. Nanonets Invoice OCR

Platform Nanonets memungkinkan pengguna membuat model OCR dengan mudah. Pengguna dapat mengunggah data, membuat anotasi, menyetel model untuk dilakukan *training*, dan menunggu prediksi melalui UI berbasis browser tanpa menulis satupun baris kode, mengkhawatirkan GPU, atau menemukan arsitektur yang tepat untuk model pembelajaran mendalam anda.

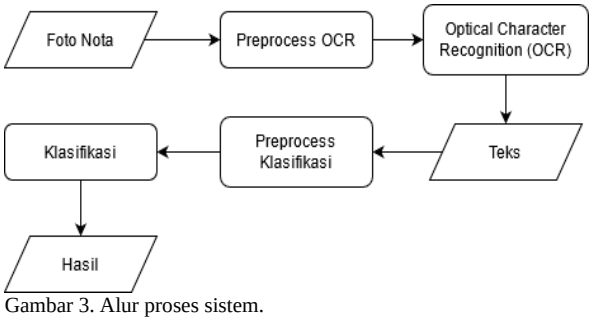
Nanonets disini adalah aplikasi sejenis yang dapat juga melakukan pembacaan nota belanja dan kemudian mengekstrak data-data yang sudah dibaca menjadi bagian-bagian yang sudah terklasifikasi.

I. Penelitian Aplikasi OCR Berbasis Web

Pada penelitian berjudul "Aplikasi OCR berbasis Web memanfaatkan Flask and Tesseract" ini, pekerjaan utama yang dilakukan adalah menjalankan program OCR sebagai layanan web. Alat paling penting untuk pendekatan ini secara luas diklasifikasikan ke dalam tiga, yaitu *front-end*, *back-end*, dan Mesin Tesseract. Teknologi *front-end* masuk ke bagian *UI/UX* dari aplikasi web. Bagian ini harus memiliki komponen termasuk HTML, CSS dan JavaScript. Meskipun framework Bootstrap dapat menjadi *framework* yang dapat diandalkan untuk *front-end* tetapi untuk kasus prototipe hanya HTML dan CSS cukup.



Gambar 2. Diagram kasus penggunaan.



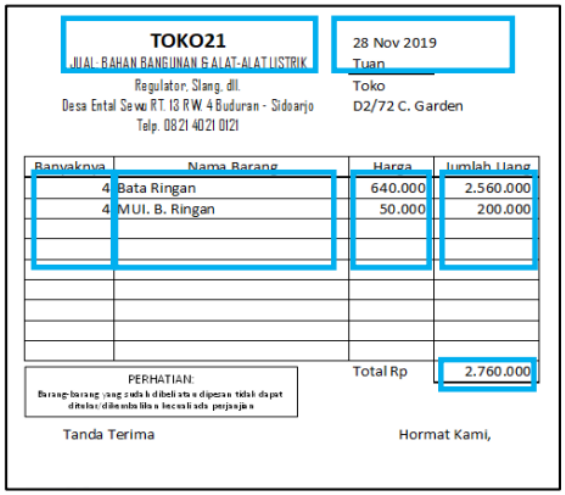
Gambar 3. Alur proses sistem.

Teknologi *back-end* yang digunakan dapat bervariasi tergantung pada kenyamanan bahasa. Paper ini menggunakan *framework* Flask yang berbasis Python sebagai back-end. Flask cepat untuk dibuat prototipenya dan mudah diimplementasikan. Setelah file inti OCR diaktifkan, file tersebut dapat digunakan dengan Flask untuk meng-host aplikasi di host lokal maupun server berbayar. Protokol yang diusulkan adalah HTTP dan HTTPS. Aplikasi dapat menerima beberapa koneksi masuk di berbagai *port*. Ini membuatnya sempurna untuk diterapkan di tingkat departemen atau universitas [4].

Paper ini digunakan oleh penulis sebagai referensi metode dalam melakukan pengembangan fitur OCR pada aplikasi Sistem Informasi Monitoring Proyek.

J. Penelitian Ekstraksi ROI

Penelitian berjudul "Metode Sederhana dan Efisien untuk Ekstraksi ROI yang Diinginkan dengan Perangkat Lunak OCR dan Program Makro" ini dirancang untuk mengekstrak nilai ROI pada gambar, dan dibuat sebagai alat pengembangan dengan menggunakan perangkat lunak OCR open-source dan program makro open-source. Proses utamanya adalah sebagai berikut: (1) menangkap wilayah nilai ROI sebagai file grafik untuk OCR, (2) mengenali teks dari gambar yang diambil oleh perangkat lunak OCR, (3) melakukan koreksi kesalahan, (4) mengekstrak nilai termasuk area, rata-rata, standar deviasi, nilai maks, dan min dari teks. Proses utama ini diulang untuk jumlah ROI. Metode sederhana dan efisien untuk ekstraksi nilai ROI dikembangkan dengan OCR open-source dan program makro. Input akurat dari berbagai angka dari ROI dapat diekstraksi dengan modul ini [5].



Gambar 4. Contoh nota beserta *region of interest*.

Tabel 2.
Contoh data dalam dataset

No	Kalimat	Label
1	toko21	nama_toko
2	jl. Pattimura	alamat
3	telp. 0851 0209 0371	telp
4	bata ringan	deskripsi
5	kepada yth	lain

Paper ini digunakan oleh penulis sebagai referensi metode dalam melakukan pengembangan fitur OCR pada aplikasi Sistem Informasi Monitoring Proyek, terutama pada bagian melakukan ekstraksi data dari wilayah Region of Interest (ROI).

III. ANALISIS DAN PERANCANGAN

A. Analisis Permasalahan

Permasalahan utama yang diangkat dalam Tugas Akhir ini adalah bagaimana membangun Sistem Informasi Monitoring Proyek (SIMP) berbasis web dengan OCR. Kegiatan pengelolaan keuangan perusahaan membutuhkan perhatian, terutama dalam hal teknis pelaksanaannya agar dapat diimplementasikan lebih cepat dan mudah. Metode penginputan data yang digunakan saat ini adalah metode manual. Sedangkan metode penginputanan data secara manual merupakan sebuah permasalahan dimana metode manual ini bisa memakan waktu yang lama.

Untuk mengatasi masalah tersebut, penulis akan mengimplementasikan pembuatan aplikasi yakni Sistem Informasi Monitoring Proyek (SIMP) untuk memudahkan pengguna dalam melakukan input data dengan mengimplementasikan OCR pada web.

Penelitian ini berfokus menitikberatkan pada Optical Character Recognition dalam proses penginputan data. Sedangkan proses detail pengenalan teks disajikan dalam proses bisnis antara aktor dengan sistem.

B. Deskripsi Umum Sistem

Aplikasi ini bernama Sistem Informasi Monitoring Proyek. Sistem Informasi Monitoring Proyek ini sebelumnya telah dikembangkan pada saat penulis melakukan Kerja Praktik. Pada tugas akhir ini, terdapat fitur yang akan ditambahkan

```
def find_score(arr, angle):
    return score

def convert_bin(filename):
    img = Image.open(filename)
    # convert to binary
    return fix_skew(img, best_angle)
def fix_skew(bin_img, best_angle):
    # Rotasi gambar dari hasil penilaian find_score
    return img

def ocr_core(filename):

    # Scan gambar ke string
    results = pytesseract.image_to_string(image, lang='ind+eng')

    # Data dipisah berdasarkan line
    hasil = results.split("\n")

    return check_first(hasil)
```

Gambar 5. Implementasi OCR.

```
def preprocess(hasil):
    for i in range(0, len(hasil)):
        # Membuat hasil menjadi lowercase
        # Memfilter hasil
    return hasil
def check_words(hasil):
    # Menghitung jumlah char
    # Jika char kurang dari 5
    if total_c<5:
        return False
    elif re.search("subtotal|invoice|nota", hasil):
        return False
    elif re.search("total", hasil):
        return "End"
    return True
def check_date(hasil):
    for h in hasil:
        # cari tanggal
        tanggal = re.search(r'\d{1,2}-\d{1,2}-\d{4}', h)
        if tanggal:
            tgl = datetime.strptime(tanggal.group(), '%d %m %Y'
).date()
            return tgl
    return date.today()
def check_first(hasil):
    tanggal=check_date(hasil)
    hasil = preprocess(hasil)
    for i in range(0, len(hasil)):
        flag=False
        # Cek jika hasil = angka setelah dilakukan check_num
            # Setelah ditemukan total, semua data setelahnya di
masukan daftar hapus
        # Dilakukan check untuk memfilter data
    # Hapus semua data yang berada di daftar hapus
    if line:
    return classification(hasil, jumlah, tanggal)
```

Gambar 6. Implementasi pra-klasifikasi.

pada aplikasi existing, yaitu OCR yang bisa digunakan untuk membaca nota dan kemudian langsung memasukannya ke dalam input field di dalam sistem dengan harapan user tidak perlu mengetiknya lagi.

Perbedaan antara Sistem Informasi Monitoring Proyek yang akan dikembangkan dengan versi sebelumnya dapat dilihat dari Tabel 1.

C. Perancangan Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak berisi representasi kebutuhan perangkat lunak hasil analisis dalam bentuk kebutuhan fungsional, diagram kasus penggunaan, dan diagram aktivitas (Gambar 2).

D. Perancangan Basis Data

Dalam membangun Sistem Informasi Monitoring Proyek dibutuhkan sebuah basis data untuk menyimpan data-data yang diperlukan. Desain basis data yang dibuat akan diimplementasikan menggunakan SQLite.

E. Perancangan Proses

Pada tahap perancangan proses, penulis merancang proses untuk pembacaan dan klasifikasi foto nota terdiri dari tahap tahap seperti dapat dilihat pada Gambar 3.

Pada penelitian ini, dilakukan proses prediksi data hasil ekstraksi foto nota. Pertama, akan dilakukan *preprocessing* pada foto nota. Pada tahap ini, dilakukan pengubahan warna foto menjadi hitam putih dan juga pengecekan apakah foto

```
def check_num(hasil):
    # Filter data berupa angka
    match = re.findall(r"[0-9]+", hasil)

    if match:
        return match
    return '0'
```

Gambar 7. Implementasi pasca-klasifikasi.

```
def classification(hasil, jumlah, tanggal):
    dataset=pd.read_csv('static/dataset.csv', delimiter=";")
    count_vect = CountVectorizer()
    tfidf_transformer = TfidfTransformer()
    text_clf_svm = Pipeline([('vect', CountVectorizer()),('tfid
f', TfidfTransformer()),('clf-svm', SGDClassifier())])
    predicted_svm = text_clf_svm.predict(hasil)

    predicted=[]
    jumlah2=0
    for i in range(0, predicted_svm.size):
        if predicted_svm[i]=="deskripsi":
            # Cek harga total
            # Cek harga satuan
            # Cek jumlah barang/qty

        predicted.append({
            "qty": harga[0],
            "kalimat": hasil[i].title(),
            "harga1": harga1[-1],
            "harga": harga[-1],
            "label": predicted_svm[i]
        })
    return predicted
```

Gambar 8. Implementasi klasifikasi.

Tabel 3.
Hasil uji kasus penggunaan

No	Kebutuhan Fungsional	Berhasil
1	Mengelola data proyek	✓
2	Mengelola data subkategori	✓
3	Mengelola data pemasukan	✓
4	Mengelola data pengeluaran	✓
5	Mengelola data persiapan	✓
6	Mengelola data arsitektur	✓
7	Mengelola data struktur	✓
8	Mengelola data MEP	✓

miring. Jika ditemukan bahwa foto miring, maka sistem akan melakukan rotasi pada foto nota. Hal tersebut dilakukan untuk membantu meningkatkan akurasi OCR. Setelah semua proses tersebut selesai, maka masuk ke tahap ekstrak data yang akan dilakukan oleh OCR. Keluaran dari tahap ini berupa teks yang kemudian akan dilakukan *preprocessing* lagi sebelum dilakukan klasifikasi. Untuk membantu dalam mengklasifikasikan data, maka penulis juga melakukan identifikasi struktur nota.

Berdasarkan identifikasi struktur nota tersebut, maka dapat disimpulkan jika data yang akan diekstrak terbagi menjadi 7 *Region of Interest*. 7 ROI yang akan diakomodasi ditampilkan pada Gambar 4. Selanjutnya masuk ke proses klasifikasi data terdapat 3 tahapan utama, yaitu tahapan pra-klasifikasi, klasifikasi dengan SVM, dan pasca-klasifikasi.

Proses pertama pada tahapan pra-klasifikasi adalah sistem akan melakukan pengecekan jika terdapat format tanggal yang sesuai menggunakan bantuan *Regex*. Setelah itu adalah pengubahan kata menjadi dalam bentuk *lowercase*, hal tersebut dilakukan karena di dalam dataset huruf besar dan kecil tidak berpengaruh terhadap hasil. Oleh karena itu teks input diubah menjadi dalam bentuk huruf kecil semua. Setelah itu terdapat proses pembuangan karakter-karakter yang dianggap tidak dibutuhkan dan mengganggu dalam proses klasifikasi seperti |, @, dan karakter lainnya. Teks yang sudah diproses tersebut kemudian akan dilakukan pengecekan kata, dimana sistem akan melakukan filter jika terdapat kata-kata yang tidak diperlukan maka akan dihapus.

Tabel 4.
Hasil uji OCR dan klasifikasi nota variasi

Gambar	*Run Time*	Hasil
1	34,119 s	2 Data salah baca 3 Data salah klasifikasi
2	1,527 s	2 Data salah baca 2 Data salah klasifikasi
3	1,160 s	3 Data salah klasifikasi
4	1,267 s	1 Data salah baca
5	5,251 s	3 Data salah baca 1 Data salah klasifikasi
6	2,127 s	1 Data salah baca 2 Data salah klasifikasi
Rata-rata	7,57 s	1,5 Data salah baca 1,83 Data salah klasifikasi

Tahapan selanjutnya adalah tahapan klasifikasi. Sebagai data training, penulis menggunakan dataset yang proses pembuatannya dilakukan secara manual. Dataset tersebut penulis ambil berdasarkan data dari beberapa nota yang sudah penulis dapatkan. Total terdapat 176 baris data yang terbagi menjadi 5 label, yaitu nama_toko, alamat, telp, deskripsi yang berisi nama-nama barang, dan lain (Tabel 2). Kelima label tersebut digunakan dengan mempertimbangkan bahwa setiap data dalam label tersebut tidak memiliki format pasti dan label dibuat sebanyak mungkin supaya sistem bisa dengan lebih mudah dalam melakukan klasifikasi.

Proses training dilakukan sebelum sistem melakukan klasifikasi dengan menggunakan dataset yang sudah dijabarkan sebelumnya. Sebelum menjalankan algoritma, data dikonversi menjadi vector numerik menggunakan *CountVectorizer*. Kemudian ditambahkan juga *TfidfTransformer* untuk menghindari bobot nilai lebih tinggi pada kalimat panjang. Klasifikasi dilakukan dengan menggunakan algoritma SVM menggunakan *SGDClassifier*.

Tahapan selanjutnya adalah tahapan pasca-klasifikasi. Pada bagian ini, hasil dari proses klasifikasi sebelumnya akan dilakukan pengecekan lebih lanjut lagi. Jika terdapat hasil klasifikasi dengan label “deskripsi”, maka sistem akan melakukan pengecekan jika terdapat angka yang terletak pada akhir baris setelah barang, maka angka tersebut diekstrak menjadi harga subtotal. Jika ditemukan set angka lagi di sebelah kiri hasil sebelumnya, maka angka tersebut diekstrak menjadi harga satuan. Jika ditemukan angka pada awal baris, maka angka tersebut akan diekstrak menjadi banyak barang.

IV. IMPLEMENTASI

A. Implementasi OCR

Bagian ini berfungsi bagi untuk melakukan proses rekognisi tulisan yang ada pada foto nota. Pada bagian pertama, akan dijalankan fungsi *convert_bin* untuk mengubah foto nota menjadi dalam bentuk hitam putih. Foto nota tersebut kemudian akan dicek kemiringannya dengan menjalankan fungsi *find_score* (Gambar 5).

Jika dalam pengecekan ternyata ditemukan bahwa foto nota tidak lurus, maka akan dijalankan fungsi *fix_skew* untuk merotasi foto tersebut sesuai dengan rekomendasi dari fungsi Scan foto menggunakan *library pytesseract*. Keluaran dari proses tersebut adalah sebuah *string*. Hasil tersebut kemudian dilakukan pemisahan berdasarkan *line*. Implementasi dari OCR dapat dilihat pada Gambar 5.

Tabel 5.
Hasil uji OCR dan klasifikasi nota

Gambar	*Run Time*	Hasil
1	5,372 s	2 Data salah klasifikasi
2	2.606 s	1 Data salah baca
3	1,929 s	4 Data salah baca 3 Data salah klasifikasi
4	30,984 s	4 Data salah baca
Rata-rata	10,222 s	2,25 Data salah baca 1,25 Data salah klasifikasi

B. Implementasi Pra-Klasifikasi

Bagian ini berfungsi untuk memproses data yang masuk dari OCR. Pada fungsi *preprocess*, dilakukan pengubahan kata menjadi dalam bentuk *lowercase*. Setelah itu terdapat proses pembuangan karakter-karakter yang dianggap tidak dibutuhkan dan malah mengganggu dalam proses klasifikasi seperti |, @, dan karakter lainnya. Pada bagian ini juga terdapat fungsi *check_words* untuk melakukan pengecekan jumlah karakter pada setiap baris dan untuk mengecek jika pada baris tersebut terkandung kata kunci yang dibutuhkan. Selain itu, juga terdapat fungsi *check_date* yang digunakan untuk mengecek apakah di dalam baris terdapat format tanggal. Implementasi dari Pra-Klasifikasi dapat dilihat pada Gambar 6.

C. Implementasi Klasifikasi

Teks yang sudah diproses kemudian akan dilakukan klasifikasi dengan menggunakan SGDClassifier yang merupakan SVM. Setelah melakukan klasifikasi, maka akan didapatkan kalimat beserta dengan label hasil klasifikasi. Sebelum menjalankan algoritma tersebut, data perlu dikonversi menjadi vektor fitur numerik dengan menggunakan CountVectorizer. Setelah itu dijalankan TF-IDF (Term Frequency-Inverse Document Frequency) untuk mendapatkan Term Frequency dikalikan dengan frekuensi dokumen invers. Implementasi dari Pra-Klasifikasi dapat dilihat pada Gambar 6.

Pada Gambar 7, hasil dari proses klasifikasi sebelumnya akan dilakukan pengecekan lagi. Jika terdapat hasil klasifikasi dengan label “deskripsi”, maka sistem akan melakukan pengecekan angka jika terdapat angka yang terletak pada akhir baris setelah barang, maka angka tersebut diekstrak menjadi harga subtotal. Jika ditemukan set angka lagi di sebelah kiri hasil sebelumnya, maka angka tersebut diekstrak menjadi harga satuan. Jika ditemukan angka pada awal baris, maka angka tersebut akan diekstrak menjadi banyak barang Hal tersebut dijalankan menggunakan fungsi *check_num*. Hasil dari proses klasifikasi selanjutnya akan dijadikan satu objek yang terdiri dari kalimat dan label. Implementasi dari Pasca-Klasifikasi dapat dilihat pada Gambar 8.

V. UJI COBA

A. Pengujian Kasus Penggunaan

Pada bagian ini akan dibahas mengenai proses uji coba pada kasus penggunaan yang digunakan. Pengujian dilakukan dengan metode black box untuk menguji masing-masing fungsionalitas yang sudah dirancang pada sistem.

Metode black box merupakan metode pengujian perangkat lunak yang memeriksa fungsionalitas dari suatu perangkat lunak tanpa memandang struktur internalnya.

Pada proses uji coba, pengujian dilakukan dengan menjalankan serangkaian perintah terhadap sistem yang selanjutnya akan disebut sebagai kasus pengujian. Kasus pengujian ini berkorelasi dengan kasus-kasus penggunaan dan kebutuhan fungsional yang sebelumnya sudah dirancang dan dijelaskan pada Bab III. Hasil pengujian dapat dilihat pada Tabel 3.

B. Pengujian OCR dan Klasifikasi Nota Variasi

Pada subbab ini akan dibahas mengenai pengujian sistem dalam melakukan pembacaan foto nota dan klasifikasi data. Dilakukan pengujian sistem dengan uji kasus 6 foto nota variasi berdasarkan pada foto nota asli. Hasil pengujian dapat dilihat di Tabel 4.

C. Pengujian OCR dan Klasifikasi Nota

Pada subbab ini akan dibahas mengenai pengujian sistem dalam melakukan pembacaan foto nota dan klasifikasi data. Dilakukan pengujian sistem dengan uji kasus 4 foto nota asli. Uji kasus terdiri dari kombinasi input scan nota dan foto. Performa dari pengujian ini berkorelasi dengan proses pembacaan foto oleh OCR dan proses klasifikasi dengan metode yang sebelumnya sudah dirancang dan dijelaskan pada Bab III. Hasil pengujian dapat dilihat di Tabel 5.

D. Evaluasi

Berdasarkan 6 kali uji coba pada foto nota variasi diperoleh rata-rata waktu yang diperlukan untuk menampilkan hasil pengenalan dan klasifikasi adalah 7,57 detik. Selain itu, didapat hasil evaluasi dengan rata-rata kesalahan adalah 1,5 dalam pembacaan dan 1,83 dalam klasifikasi. Hal tersebut dapat diartikan juga jika dalam setiap melakukan scan foto nota terdapat paling tidak 1 kesalahan pembacaan maupun klasifikasi. Hal ini tidak bagus.

Kemudian berdasarkan 4 kali uji coba pada foto nota asli diperoleh rata-rata waktu yang diperlukan untuk menampilkan hasil pengenalan dan klasifikasi adalah 10,222 detik. Hal ini relatif lama. Selain itu, didapat hasil evaluasi dengan rata rata kesalahan adalah 2,25 dalam pembacaan dan 1,25 dalam klasifikasi. Hal tersebut dapat diartikan juga jika dalam setiap melakukan scan foto nota terdapat paling tidak 2 kesalahan pembacaan dan 1 kesalahan klasifikasi. Hal ini tetap tidak bagus.

VI. KESIMPULAN

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut: (1) Informasi penting nota pembelian berhasil diidentifikasi berdasarkan struktur nota dan klasifikasi menggunakan Support Vector Machine. (2) Pencocokan antara label klasifikasi dan input field web memungkinkan data bisa dibaca web.

DAFTAR PUSTAKA

[1] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[2] C. C. Tappert and S. H. Cha, “English language handwriting recognition interfaces,” *Text Entry Syst.*, pp. 123–137, 2007, doi: 10.1016/B978-012373591-1/50006-1.

[3] P. M. Manwatkar and K. R. Singh, “A Technical Review on Text Recognition From Images,” in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, 2015, pp. 1–5.

[4] S. Sivkov *et al.*, “The algorithm development for operation of a computer vision system via the OpenCV library,” *Procedia Comput. Sci.*, vol. 169, pp. 662–667, 2020.

[5] Y. H. Lee, E. H. Park, and J.-S. Suh, “Simple and efficient method for region of interest value extraction from picture archiving and communication system viewer with optical character recognition software and macro program,” *Acad. Radiol.*, vol. 22, no. 1, pp. 113–116, 2015.