Jurnal Teknologi Sistem Informasi dan Aplikasi **ISSN: 2654-3788**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2654-4229**
Vol. 6, No. 2, April 2023 (63-71) **DOI: 10.32493/jtsi.v6i2.29235**

Implementasi Algoritma LSTM pada Aplikasi Optical Character Recognition Berbasis Website Menggunakan Tesseract OCR

Alpha Fausta Ikrar Setyadi[1], Yeremia Alfa Susetyo[2]

[1, 2]Teknik Informatika, Universitas Kristen Satya Wacana, Jl. Dr. O. Notohamidjojo No. 1-10, Salatiga, Jawa Tengah, Indonesia, 50715
e-mail: [1]672019219@student.uksw.edu, [2]yeremia.alfa@uksw.edu

Submitted Date: March 15th, 2023 Reviewed Date: March 24th, 2023
Revised Date: March 27th, 2023 Accepted Date: March 30th, 2023

Abstract

The practicality of digital document processing has made various companies and organizations switch physical documents to digital. However, the process of extracting data from physical documents manually requires effort and is prone to input errors due to human error. Optical Character Recognition (OCR) technology can be a solution to this problem. OCR is used to recognize letters or characters in an image, and then stored into text data on a computer. In this research, the implementation of OCR technology on a web-based application with Long Short-Term Memory method. Based on accuracy testing, the average error value at the character level is 6.56% and at the word level is 9.98%. From the results obtained, it shows that the application of OCR technology with Long Short-Term Memory method on web applications can be the right solution in the process of extracting data from physical document.

Keywords: Optical Character Recognition; Long Short-Term Memory; Tesseract

Abstrak

Pengolahan dokumen digital yang lebih praktis membuat berbagai instansi dan organisasi beralih dokumen fisik menjadi digital. Namun proses ekstraksi data dari dokumen fisik secara manual membutuhkan usaha yang tidak mudah dan rentan akan terjadinya kesalahan input akibat human error. Teknologi Optical Character Recognition (OCR) dapat menjadi solusi dari permasalahan ini. OCR digunakan untuk mengenali huruf atau karakter yang ada pada suatu gambar, untuk kemudian disimpan menjadi data teks pada komputer. Pada penelitian ini, dilakukan implementasi teknologi OCR pada aplikasi berbasis website dengan metode Long Short-Term Memory. Berdasarkan pengujian akurasi diperoleh rata-rata nilai error pada tingkat karakter sebesar 6.56% dan pada tingkat kata sebesar 9,98%. Dari hasil yang didapat menunjukkan bahwa penerapan teknologi OCR dengan metode Long Short-Term Memory pada aplikasi website dapat menjadi solusi yang tepat dalam proses ekstraksi data dari dokumen fisik.

Kata kunci: Pengenalan Karakter Optik; Long Short-Term Memory; Tesseract

1. Pendahuluan

Dewasa ini banyak instansi atau organisasi yang memilih untuk melakukan penyimpanan dokumen dalam bentuk digital. Hal ini dikarenakan proses ekstraksi dan penyuntingan informasi dari dokumen cenderung lebih mudah dilakukan dalam bentuk digital, daripada dalam bentuk fisik. Selain itu, penyimpanan dokumen dalam bentuk digital juga membutuhkan tempat dan biaya yang jauh lebih kecil daripada dokumen fisik. Untuk melakukan pengolahan dari dokumen fisik ke digital membutuhkan usaha yang tidak mudah. Proses memasukkan informasi dari dokumen fisik secara manual tentu saja akan membutuhkan waktu yang lama, selain itu juga memiliki resiko yang tinggi terhadap terjadinya kesalahan input data akibat dari human error.

Permasalahan tersebut dapat diselesaikan dengan menggunakan teknologi Optical Character Recognition (OCR). Teknologi ini bekerja dengan

mengekstrak karakter atau tulisan yang terdapat pada citra atau gambar. Dalam hal ini, teknologi OCR digunakan untuk memindai informasi dari bentuk gambar menjadi bentuk teks tanpa harus memasukkan secara manual. Sehingga dapat dilakukan pengolahan data secara digital dengan lebih mudah. Proses input data dengan menggunakan OCR juga dapat mengurangi terjadinya kesalahan karena memiliki tingkat akurasi yang tinggi. Selain itu, penyimpanan dokumen dalam bentuk teks cenderung membutuhkan ruang yang lebih kecil daripada penyimpanan dalam bentuk gambar (Santoso et al., 2020).

Tesseract merupakan salah satu library OCR yang bersifat open source dan paling banyak digunakan saat ini. Saat ini, Tesseract dapat mengenali lebih dari 100 bahasa dan masih dapat dilatih untuk dapat mengenali jenis bahasa yang lain. Namun, Tesseract masih memiliki kekurangan, salah satunya adalah hasil dari OCR sangat dipengaruhi oleh kualitas dari masukan gambar (Idrees dan Hassani, 2021). Sehingga untuk mendapatkan hasil yang maksimal diperlukan preprocessing gambar terlebih dahulu.

Sejak versi keempat, Tesseract menerapkan algoritma Long Short-Term Memory (LSTM) pada model untuk meningkatkan performanya, baik dari sisi kecepatan maupun akurasi. LSTM merupakan salah satu bentuk dari Recurrent Neural Network (RNN). Arsitektur LSTM dikembangkan untuk mengatasi kekurangan dari algoritma RNN konvensional, yaitu permasalahan vanishing gradient. Pada Tesseract, algoritma LSTM lebih difokuskan untuk mengenali garis pada karakter tulisan dalam gambar. Menurut penelitian, implementasi LSTM pada Tesseract mampu meningkatkan akurasi pengenalan karakter dibandingkan dengan versi sebelumnya (Singh & Bhushan, 2019).

Pada penelitian yang dilakukan, penulis menggunakan algoritma Long Short-Term Memory pada aplikasi OCR berbasis website. Pada penelitian ini difokuskan untuk melakukan penerapan algoritma LSTM menggunakan library Tesseract. Sehingga proses pembuatan aplikasi dilakukan dengan menggunakan library Tesseract LSTM saja, tanpa Tesseract Legacy Engine. Teknologi OCR diimplementasikan dalam bentuk aplikasi web, sehingga dapat digunakan dengan mudah dan dapat diakses dari berbagai jenis perangkat karena hanya membutuhkan web browser saja. Aplikasi OCR dirancang agar dapat mudah digunakan, pengguna hanya perlu mengunggah foto atau dokumen kemudian sistem akan mengembalikan hasil scan OCR bentuk teks. Melalui aplikasi yang dibuat diharapkan dapat membantu proses ekstraksi data dari dokumen fisik, sehingga dapat mempermudah untuk melakukan penyimpanan, pencarian, maupun penyuntingan dokumen.

2. Penelitian Terdahulu

Penerapan OCR pada aplikasi website dapat mempermudah pihak yang memerlukan proses input data KTP tanpa harus memasukkan data secara manual (Toha dan Triayudi, 2022). Proses input data hanya perlu dilakukan dengan mengunggah gambar KTP ke situs web, kemudian sistem akan mengekstrak data yang terdapat pada KTP. Penelitan tersebut memiliki kesamaan kasus dengan penelitian yang dilakukan penulis, yaitu dengan mengimplementasikan teknologi OCR pada aplikasi web. Dengan turut mengimplementasikan OCR pada aplikasi website, diharapkan penelitian ini dapat membantu mempermudah proses input dari berkas fisik ke digital tanpa harus memasukkan secara manual.

Tesseract LSTM pada aplikasi OCR merupakan salah satu solusi yang tepat untuk proses input data. Pada penelitian yang dilakukan oleh Lestari, pembacaan 19 atribut pada e-KTP menghasilkan akurasi sebesar 98,42% dan pengujian kesalahan pembacaan atribut e-KTP menghasilkan akurasi sebesar 93,56%. Dari jenis pengujian akurasi yang dilakukan, didapat nilai akurasi secara keseluruhan sebesar 92.1% (Lestari dan Pratama, 2022). Algoritma yang digunakan pada penelitian tersebut memiliki kesamaan dengan penelitian ini, yaitu algoritma Long Short-Term Memory dengan menggunakan Tesseract OCR. Dari tingginya persentase akurasi yang didapat dalam penelitian tersebut menjadi faktor pendorong bagi peneliti untuk menggunakan algoritma yang sama dalam penelitian ini. Sehingga diharapkan aplikasi yang dirancang dalam penelitian ini dapat memperoleh nilai akurasi yang baik.

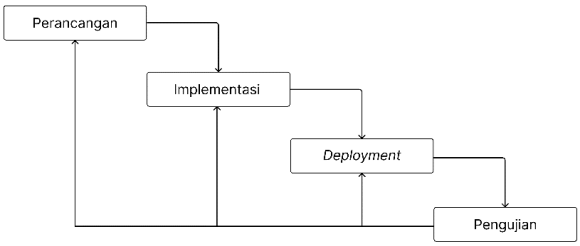
Pada penelitian yang dilakukan oleh Andreas, dilakukan implementasi Tesseract pada aplikasi mobile untuk mengambil karakter yang dicetak pada nota. Pada penelitian ini dihasilkan

kesimpulan bahwa Tesseract sangat bergantung pada tahapan preprocessing gambar sebelum melakukan proses OCR (Andreas et al., 2020). Dari kekurangan pada Tesseract OCR yang didapatkan dalam penelitian tersebut, maka pada penelitian ini dilakukan tahap preprocessing gambar terlebih dahulu sebelum dilakukan proses OCR. Sehingga didapatkan hasil OCR yang lebih maksimal.

Menurut penelitian yang dilakukan oleh Ujwala dan Sumanthi, implementasi teknologi Optical Character Recognition dengan menggunakan algoritma LSTM mampu mengenali karakter huruf pada gambar dengan efektif (Ujwala dan Sumathi, 2019). Dari penelitian didapatkan hasil bahwa algoritma LSTM pada Tesseract OCR mampu mengenali karakter dengan berbagai jenis font dan orientasi yang berbeda-beda. Kelebihan dari LSTM Tesseract OCR yang didapatkan dalam penelitian tersebut mendorong peneliti untuk turut menggunakan algoritma dan library yang sama untuk melakukan pendeteksian teks pada dokumen.

3. Metode Penelitian

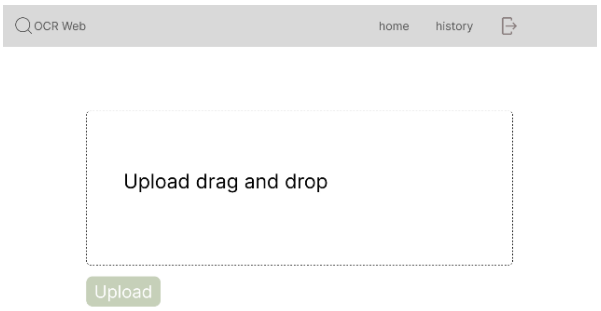
Sumber permasalahan pada penelitian ini adalah bagaimana mengimplementasikan algoritma Long Short-Term Memory menggunakan Tesseract OCR pada aplikasi website agar dapat dengan mudah digunakan oleh pengguna. Melalui penelitian ini diharapkan dapat memberikan solusi atau jawaban atas permasalahan tersebut. Gambar 4 merupakan ilustrasi langkah-langkah yang dilakukan dalam menyelesaikan penelitian ini.



Gambar 1. Tahapan pembuatan aplikasi

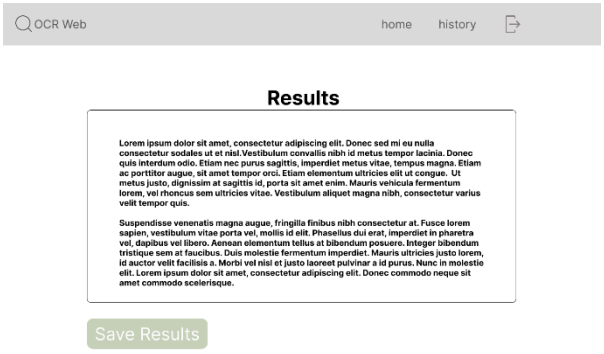
Perancangan

Tahapan pertama dalam membuat aplikasi OCR adalah perancangan. Tahap ini dilakukan untuk menentukan rancangan arsitektur, desain tampilan aplikasi, dan flow chart dari aplikasi. Pada rancangan arsitektur, aplikasi website dijalankan pada Google Compute Engine. Hal ini dilakukan karena Tesseract membutuhkan virtual machine untuk dapat berjalan pada layanan cloud. Database yang digunakan pada aplikasi ini adalah MongoDB. Hal ini dikarenakan performa MongoDB yang cepat dan mudah untuk digunakan. Selain itu, MongoDB menggunakan format binary JSON yang membuat pengguna MongoDB tidak perlu merancang struktur tabel seperti pada SQL dan struktur data akan terbentuk secara otomatis saat melakukan proses insert (Firdaus et al., 2021).



Gambar 2. Wireframe halaman upload

Pada tahap ini juga dilakukan perancangan tampilan aplikasi. Perancangan ini dilakukan untuk membuat tampilan aplikasi yang mudah dipahami oleh pengguna. Rancangan wireframe dari aplikasi website OCR ditunjukkan pada gambar 2 dan 3.



Gambar 3. Wireframe halaman hasil

Implementasi

Setelah perancangan, langkah selanjutnya adalah tahapan implementasi. Pada tahapan ini, dilakukan proses pengkodean program baik front-end maupun back-end. Proses implementasi dari aplikasi OCR website dilakukan menggunakan bahasa pemrograman Python dan framework Flask. Peneliti memilih bahasa pemrograman Python karena memiliki sintaks yang lebih singkat, sehingga dapat mempercepat dan mempermudah

proses pengkodean. Sedangkan framework Flask dipilih karena ringan dan tidak memiliki terlalu banyak built-in library. Proses implementasi akan menggunakan library pytesseract, yang merupakan wrapper Tesseract untuk bahasa pemrograman Python. Library ini memungkinkan pengembang untuk dapat langsung berinteraksi dengan Tesseract menggunakan bahasa pemrograman Python.

Deployment

Setelah proses pengkodean aplikasi selesai, dilakukan proses deployment aplikasi dari komputer lokal ke layanan cloud. Layanan cloud yang digunakan dalam tahap implementasi adalah Google Compute Engine (GCE), sedangkan database yang digunakan adalah layanan MongoDB Atlas. Proses deployment diawali dengan membuat VM instance. Karena hanya untuk kebutuhan penelitian, penulis memilih menggunakan tipe N2-Standard-2 dengan sistem operasi Debian. Setelah membuat instance, dilanjutkan dengan melakukan clone sumber kode yang sebelumnya telah disimpan pada repositori github. Kemudian dilakukan proses pemasangan library Tesseract dan dependencies yang di-perlukan untuk menjalankan aplikasi OCR. Hal lain yang perlu dipersiapkan adalah database. Proses pembuatan database diawali dengan membuat cluster MongoDB Atlas. Untuk menghubungkan aplikasi dengan database, connection string yang sebelumnya terhubung pada MongoDB lokal diubah agar menjadi terhubung degan MongoDB Atlas. Hal terakhir yang perlu dilakukan adalah menghubungkan MongoDB Atlas ke layanan GCE. Karena cluster yang digunakan adalah free tier, maka proses koneksi dilakukan dengan membuka akses pada IP instance Compute Engine.

Pengujian

Tahapan terakhir yang dilakukan adalah dengan melakukan pengujian untuk memastikan sistem dapat berjalan dengan baik. Pada tahapan ini dilakukan pengujian akurasi OCR menggunakan metode evaluasi Character Error Rate (CER) dan Word Error Rate (WER). CER didasarkan pada konsep jarak Levenshtein, dimana dilakukan perhitungan jumlah kesalahan pada tingkat karakter dari output OCR (Tsimpiris et al., 2022). Perhitungan CER dilakukan dengan persamaan berikut.

$$\mathrm{CER} = \frac{S+D+I}{N} \quad (1)$$

S : Jumlah substitusi karakter
D : Jumlah karakter yang terhapus
I : Jumlah karakter yang tersisip
N : Jumlah karakter sebenarnya

Perhitungan WER kurang lebih sama dengan CER, hanya saja dilakukan pada level kata. Hasil perhitungan WER sangat berkorelasi dengan CER, namun biasanya WER cenderung memiliki nilai yang lebih tinggi daripada CER. Berikut merupakan bentuk persamaan dari WER.

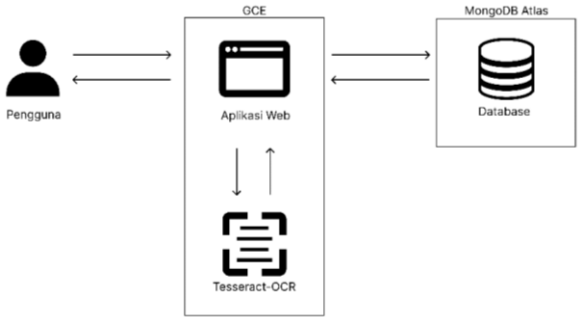
$$\mathrm{WER} = \frac{S_w+D_w+I_w}{N_w} \quad (2)$$

Sw : Jumlah substitusi karakter
Dw : Jumlah karakter yang terhapus
Iw : Jumlah karakter yang tersisip
Nw : Jumlah karakter sebenarnya

4. Hasil dan Pembahasan

Implementasi aplikasi

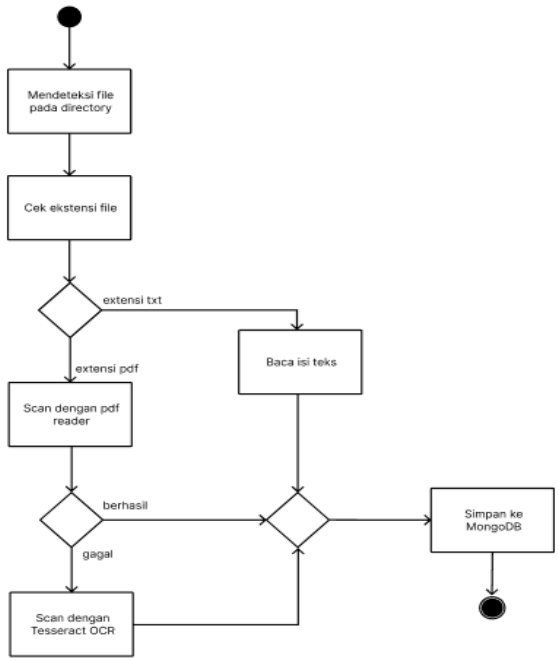
Pada penelitian ini dilakukan implementasi teknologi Optical Character Recognition (OCR) pada aplikasi berbasis web. Pemilihan platform website dikarenakan penggunaannya yang cukup mudah. Pengguna hanya memerlukan aplikasi browser dan koneksi internet untuk dapat menggunakan aplikasi OCR.



Gambar 4. Arsitektur aplikasi

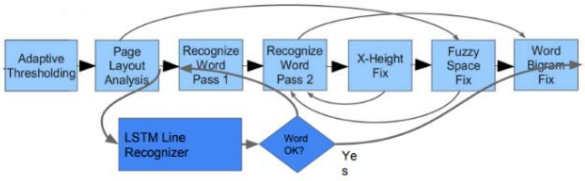
Gambar 4 merupakan ilustrasi arsitektur dari aplikasi yang telah dibuat. Aplikasi OCR berbasis website ini dibuat dengan menggunakan library Tesseract dan bahasa pemrograman Python. Karena memerlukan virtual machine untuk menjalankan library Tesseract, maka aplikasi yang telah dibuat tidak dapat dijalankan pada layanan

hosting biasa. Hasil aplikasi dijalankan pada layanan cloud computing, dengan menggunakan Google Compute Engine (GCE). Pada bagian database, peneliti menggunakan MongoDB Atlas sebagai cloud database.



Gambar 5. Alur kerja aplikasi

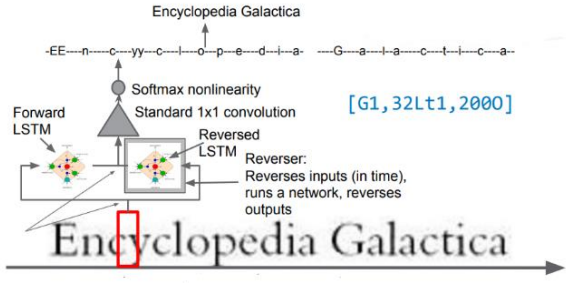
Alur kerja dari aplikasi diilustrasikan pada gambar 5. Dimulai dari user mengunggah gambar pada situs web yang telah disediakan. Kemudian sistem akan memeriksa ekstensi file apakah sudah sesuai dengan ekstensi yang ditentukan dan untuk menentukan apakah perlu dilakukan konversi tipe file. Jika file bertipe gambar, maka proses akan langsung dilanjutkan. Apabila file bertipe pdf, maka akan dilakukan konversi file terlebih dahulu untuk kemudian dilakukan proses selanjutnya. Jika bukan keduanya, makan sistem akan memberikan peringatan pada pengguna bahwa tipe file yang diunggah belum sesuai. Tahap selanjutnya adalah preprocessing. Tahap ini dilakukan untuk menyesuaikan input gambar agar Tesseract mampu mengenali karakter pada gambar dengan lebih baik. Terdapat beberapa tahapan preprocessing yang dilakukan pada aplikasi ini, antara lain grayscaling, binarization, noise reduction, normalization, thinning dan skeletonization. Grayscaling dilakukan untuk mengubah citra berwarna menjadi keabuan. Proses ini dilakukan dengan mengatur tingkat keabuan berdasarkan rata-rata tingkat intensitas piksel warna RGB (Red Green Blue) (Mursari dan Wibowo, 2021). Binarization atau binerisasi adalah proses yang dilakukan untuk mendapatkan citra yang lebih jelas dan kontras sehingga dapat mempermudah proses selanjutnya (Nurhaliza et al., 2022). Berdasarkan penelitian yang dilakukan oleh Bukhari, citra yang telah dilakukan proses grayscaling dan binarization pada proses pelatihan arsitektur LSTM dapat menghasilkan peningkatan akurasi yang signifikan pada model (Bukhari et al., 2018). Noise reduction merupakan proses untuk mengurangi noise yang ada pada gambar. Pada penelitian ini, proses noise reduction dilakukan dengan menggunakan teknik mean filtering. Teknik ini dilakukan untuk memberikan efek smoothing, dimana titik pada suatu piksel akan digantikan dengan nilai rata-rata dari matriks di sekitarnya (Yuwono, 2010). Normalization atau normalisasi dilakukan untuk menyesuaikan input data citra dengan data citra yang telah dikenali model (Hartanto et al., 2014). Setelah tahap preprocessing selesai, dilanjutkan dengan tahap scan OCR. Tahap ini dilakukan dengan menggunakan Tesseract LSTM.



Gambar 6. Cara kerja Tesseract OCR (Sumber : https://Tesseract OCR.github.io/docs/das_tutorial2016/2ArchitectureAndDataStructures.pdf)

Gambar 6 menunjukkan cara kerja Tesseract OCR untuk melakukan pengenalan karakter. Proses dimulai dengan melakukan konversi gambar menjadi bentuk binary image menggunakan adaptive thresholding. Kemudian dilanjutkan dengan analisa tata letak halaman. Tesseract menggunakan Page Layout Analysis untuk mengenali tata letak halaman yang dilakukan dengan mengenali garis dan gambar pada halaman, kemudian dilanjutkan dengan mencari kemungkinan pembagian kolom pada halaman, dan akhirnya model mendeteksi blok teks pada halaman. Proses selanjutnya adalah pengenalan karakter. Tesseract menggunakan dua tahap pengenalan karakter untuk mendapatkan hasil yang maksimal. Tahapan ini dikenal dengan adaptive recognition (Lestari dan Mulyana, 2022). Hasil pengenalan karakter dari tahap yang pertama

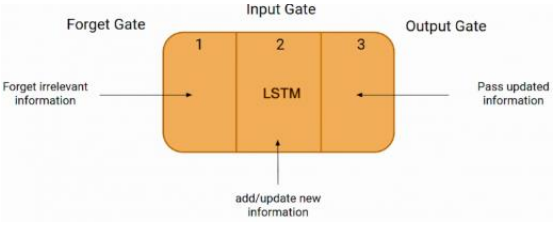
digunakan sebagai data training pada classifier untuk meningkatkan akurasi model. Pada proses pengenalan karakter, juga dilakukan perbaikan x-height untuk menentukan jarak antara baseline dan meanline pada karakter (Smith, 2007).



Gambar 7. LSTM pada Tesseract OCR
(Sumber: https://Tesseract OCR.github.io/docs/das_tutorial2016/6ModernizationEfforts.pdf)

Tesseract memiliki metode lain dalam melakukan pengenalan kata, yaitu dengan menggunakan algoritma Long Short-Term Memory untuk mengenali character sequence dari teks pada gambar. Gambar 7 merupakan ilustrasi bagaimana cara kerja dari algoritma LSTM pada Tesseract OCR dalam mengenali pola dari karakter pada gambar teks oleh Tesseract.

Long Short-Term Memory atau biasa disingkat LSTM merupakan salah satu tipe dari Recurrent Neural Network (RNN) yang sering digunakan dalam menyelesaikan permasalahan prediksi data sekuensial. LSTM dikembangkan untuk mengatasi permasalahan yang sering ditemui pada algoritma RNN konvensional, yaitu vanishing gradient. Permasalahan ini disebabkan oleh gradien yang semakin mengecil hingga layer terakhir yang membuat nilai bobot tidak pernah berubah sehingga menyebabkan kondisi menjadi konvergen.



Gambar 8. Bagian LSTM
(Sumber: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-shortterm-memory-lstm/)

Gambar 8 merupakan ilustrasi dari sel LSTM. Secara umum LSTM dibagi menjadi tiga bagian, yaitu forget gate, input gate, dan output gate. Forget gate berfungsi untuk memilih informasi dari timestamp sebelumnya akan diingat dan yang tidak relevan akan dilupakan. Input gate berfungsi untuk mempelajari informasi baru dari masukan ke dalam sel. Sedangkan output gate berfungsi untuk mengirimkan informasi terbaru dari timestamp saat ini ke timestamp berikutnya.

Tabel 1. Pseudocode algoritma LSTM

Algoritma	: Long Short-Term Memory (LSTM)
Input	: X (data sekuensial)
Output	: h_t, c_t

```
Inisialisasi parameter: Wf, Wi, Wc, Wo, Uf, Ui, Uc, Uo, bf, bi, bc, bo, ht, ct

Untuk setiap xt pada data sekuensial X:
    # menghitung gerbang forget
    ft = sigmoid(Wf * xt + Uf * ht-1 + bf)
    # gerbang input
    it = sigmoid(Wi * xt + Ui * ht-1 + bi)
    # kandidat cell state
    ct~ = tanh(Wc * xt + Uc * ht-1 + bc)
    ct = ft * ct + it * ct~ # cell state
    # gerbang output
    ot = sigmoid(Wo * xt + Uo * ht-1 + bo)
    ht = ot * tanh(ct) #output
akhir perulangan
return ht, ct
```

Tabel 1 merupakan pseudocode yang menggambarkan program algoritma LSTM. Tahap pertama dari pemrosesan data menggunakan LSTM dimulai melalui komponen forget gate. Pada bagian ini, informasi yang kurang relevan akan dihapus menggunakan fungsi sigmoid. Forget gate akan menghasilkan nilai antara 0 dengan 1. Nilai 0 berarti informasi dari timestamp sebelumnya akan dilupakan, sedangkan nilai 1 berarti informasi akan tetap diingat. Berikut merupakan formula perhitungan dari komponen forget gate.

$$f_t = \sigma(x_t * U_f + h_{t-1} * W_f) \quad (3)$$

x_t : masukan informasi pada timestamp saat ini
U_f : bobot yang terasosiasi dengan input
h_{t-1} : vektor hidden state dari timestamp sebelumnya

W_f : bobot matrix yang terasosiasi dengan hidden state

Tahap berikutnya, informasi yang masuk akan diolah pada komponen input gate. Proses ini akan memilih dan menentukan informasi yang akan diperbarui ke bagian cell state menggunakan fungsi aktivasi sigmoid. Proses ini dilakukan dengan menggunakan persamaan berikut.

$$i_t = \sigma(x_t * U_i + h_{t-1} * W_i) \quad (4)$$

x_t : input pada timestamp saat ini
U_i : bobot input matrix
h_{t-1} : vektor hidden state dari timestamp sebelumnya
W_i : bobot matrix yang terasosiasi dengan hidden state

Pada tahapan ini juga dilakukan pembentukan informasi baru menggunakan fungsi aktivasi tanh yang akan ditambahkan pada cell state. Apabila informasi baru benilai positif, nilai akan ditambahkan ke cell state. Sebaliknya jika bernilai negatif, maka cell state akan dikurangi dengan nilai informasi baru. Perhitungan proses ini dilakukan dengan persamaan berikut.

$$N_t = \tanh(x_t * U_c + h_{t-1} * W_c) \quad (5)$$

Setelah didapatkan nilai informasi baru, maka nilai dari cell state akan diperbarui dengan menggunakan persamaan berikut.

$$C_t = f_t * C_{t-1} + i_t * N_t \quad (6)$$

Tahap yang terakhir dilakukan pada bagian output gate yang memiliki bentuk hampir sama dengan gate sebelumnya. Output gate juga akan menghasilkan nilai antara 0 dengan 1, karena menggunakan fungsi aktivasi sigmoid. Fungsi persamaan dari output gate adalah sebagai berikut:

$$O_t = \sigma(x_t * U_o + h_{t-1} * W_o) \quad (7)$$

x_t : input pada timestamp saat ini
U_i : bobot output matrix
h_{t-1} : vektor hidden state dari timestamp sebelumnya
W_o : bobot matrix yang terasosiasi dengan hidden state

Kemudian untuk melakukan kalkulasi nilai hidden state perlu dilakukan perkalian antara nilai dari output gate dengan aktivasi tanh dari cell state yang telah diperbarui, seperti pada persamaan berikut.

$$H_t = O_t * \tanh(C_t) \quad (8)$$

Pengujian sistem

Pada penelitian ini dilakukan pengujian sistem sebagai evaluasi akurasi OCR dengan menggunakan perhitungan Word Error Rate dan Character Error Rate. Pengujian akurasi dilakukan untuk menilai tingkat ketepatan OCR dalam mendeteksi karakter dari masukan gambar. Pengujian ini dilakukan terhadap dua jenis dokumen, yaitu invoice, dan dokumen perjanjian dengan masing-masing 3 dokumen pada setiap jenisnya. Hal ini dilakukan untuk mengetahui tingkat akurasi dari Tesseract OCR dalam mengenali karakter pada dokumen dengan tata letak yang berbeda-beda. Karena dokumen invoice cenderung memiliki bentuk tabel, sedangkan dokumen perjanjian cenderung berbentuk paragraf panjang.

Tabel 2. Hasil evaluasi WER dan CER

Jenis Dokumen	Hasil WER	Hasil CER
Dokumen 1 (Invoice)	2.54%	0.91%
Dokumen 2 (Invoice)	13.43%	2.17%
Dokumen 3 (Invoice)	3.33%	0.37%
Dokumen 4 (Perjanjian)	13.58%	15.86%
Dokumen 5 (Perjanjian)	16.51%	14.54%
Dokumen 6 (Perjanjian)	10.54%	5.49%
Rata-rata	9.98%	6.56%

Tabel 2 merupakan hasil evaluasi menggunakan Word Error Rate dan Character Error Rate. Nilai error yang didapat menunjukkan seberapa banyak terjadi kesalahan pengenalan pada tingkat karakter dan juga kata. Dapat terlihat dari tabel, hasil perhitungan menggunakan WER memiliki nilai yang lebih tinggi daripada CER. Berdasarkan pengujian yang dilakukan, didapatkan

rata-rata evaluasi WER sebesar 9.98% dan CER 6.56%.

5. Kesimpulan

Berdasarkan pengujian hasil OCR terhadap dokumen tagihan dan perjanjian, diperoleh rata-rata nilai error pada tingkat karakter sebesar 6.56% dan pada tingkat kata sebesar 9.98%. Dokumen tagihan menunjukkan rata-rata nilai error yang lebih kecil dari dokumen perjanjian. Dari hasil tersebut dapat terlihat bahwa penerapan algoritma Long Short-Term Memory dapat memberikan hasil yang baik pada pengenalan pada tingkat karakter, namun kurang baik pada tingkat kata. Hal ini disebabkan karena adanya substitusi, interpolasi, dan eliminasi karakter pada kata. Sehingga kesalahan pada tingkat kata cenderung memiliki nilai yang lebih tinggi daripada tingkat karakter.

6. Saran

Dari penelitian yang telah dilakukan, terdapat beberapa hal yang menjadi saran untuk pengembangan pada penelitian selanjutnya, yaitu:

- Dapat dilakukan pengujian terhadap bentuk dokumen yang lebih beragam untuk mengetahui pengaruh penggunaaan algoritma LSTM untuk pengenalan karakter pada berbagai jenis dokumen.
- Dapat dilakukan implementasi teknologi OCR platform lain, seperti mobile atau progressive web application.

References

Andreas, Y., Gunadi, K., & Purbowo, A. N. (2020). Implementasi Tesseract OCR untuk Pembuatan Aplikasi Pengenalan Nota pada Android. *JURNAL INFRA*, *8*(1).

Bukhari, S. S., Francis, S., Kamath, C. N. N., & Dengel, A. (2018). An investigative analysis of different LSTM libraries for supervised and unsupervised architectures of OCR training. *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, *2018-August*, 447–452. https://doi.org/10.1109/ICFHR-2018.2018.00084

Cahyo Santoso, B., Natasya, Y., Willian, S., & Alfando, F. (2020). Tinjauan Pustaka Sistematis terhadap Basis Data MongoDB. *JII: Jurnal Inovasi Informatika Universitas Pradita*, *5*(2), 132–142.

Firdaus, A., Syamsu Kurnia, M., Shafera, T., Firdaus, W. I., Teknik, J., Politeknik, K., & Sriwijaya - Palembang, N. (2021). Implementasi Optical Character Recognition (OCR) Pada Masa Pandemi Covid-19. *Jurnal JUPITER*, *13*(2), 188–194.

Hartanto, S., Sugiharto, A., Sukmawati, D., & Endah, N. (2014). Optical Character Recognition Menggunakan Algoritma Template Matching Correlation. *Jurnal Masyarakat Informatika*, *5*(9), 1–14.

Idrees, S., & Hassani, H. (2021). Exploiting script similarities to compensate for the large amount of data in training tesseract lstm: Towards kurdish ocr. *Applied Sciences (Switzerland)*, *11*(20). https://doi.org/10.3390/app11209752

Lestari, I. N. T., & Mulyana, D. I. (2022). Implementation of Ocr (Optical Character Recognition) Using Tesseract in Detecting Character in Quotes Text Images. *Journal of Applied Engineering and Technological Science*, *4*(1), 58–63.

Lestari, S., & Fakhri Pratama, M. (2022). Penerapan Metode Long Short-Term Memory Pada Pendataan Warga Berbasis Android. *Journal of Computer System and Informatics (JoSYC)*, *3*(4), 156–161. https://doi.org/10.47065/josyc.v3i4.1951

Mursari, L. R., & Wibowo, A. (2021). The Effectiveness of Image Preprocessing on Digital Handwritten Scripts Recognition with The Implementation of OCR Tesseract. *Computer Engineering and Applications*, *10*(3).

Nurhaliza, S. S., Subali, M., Etp, L., & Rozi, D. (2022). Analisis Kinerja Optical Character Recognition untuk Membaca Dokumen Secara Otomatis. In *Seminar Nasional Teknologi Informasi dan Komunikasi STI&K (SeNTIK)* (Vol. 6, Issue 1).

Singh, J., & Bhushan, B. (2019). Real Time Indian License Plate Detection using Deep Neural Networks and Optical Character Recognition using LSTM Tesseract. *IEEE 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 347–352.

Smith, R. (2007). An Overview of the Tesseract OCR Engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, *2*, 619–633. https://doi.org/10.1109/ICDAR.2007.4376991

Toha, M. R., & Triayudi, A. (2022). Penerapan Membaca Tulisan di dalam Gambar Menggunakan Metode OCR Berbasis Website pada e-KTP. *Jurnal Sains Dan Teknologi*, *11*, 175–183. https://doi.org/10.23887/jst-undiksha.v11i1

Tsimpiris, A., Varsamis, D., & Pavlidis, G. (2022). Tesseract OCR evaluation on Greek food menus datasets. *International Journal of Computing and Optimization*, *9*(1), 13–32. https://doi.org/10.12988/ijco.2022.9829

Ujwala B S, & Sumathi K. (2019). A Novel Approach

Towards Implementation Of Optical Character Recognition Using LSTM And Adaptive Classifier. *JNNCE Journal of Engineering & Management (JJEM)*, *3*(2), 59–68. https://doi.org/10.37312/JJEM.2019.030206

Yuwono, B. (2010). Image Smoothing Menggunakan Mean Filtering, Median Filtering, Modus Filtering dan Gaussian Filtering. *Telematika : Jurnal Informatika Dan Teknologi Informasi*, *7*(1). https://doi.org/https://doi.org/10.31315/telematika.v7i1.416