ANALISIS *ENTITY MATCHING* PADA *DATASET SMARTPHONE* MENGGUNAKAN METODE S*IF, RNN, ATTENTION,* DAN *HYBRID*

Rahmat Hidayat*, Rivanda Putra, Nur Aini Rakhmawati
Departemen Sistem Informasi
Institut Teknologi Sepuluh Nopember Surabaya
Jl. Teknik Kimia, Kota Surabaya, Jawa Timur. 60117
*E-mail: rahmat.19052@mhs.its.ac.id

Abstrak: Penerapan teknologi informasi saat ini berdampak pada kecepatan dan efektivitas suatu perusahaan atau masyarakat. Perkembangan dan kecepatan data saat ini sangat krusial, perusahaan terus berupaya untuk mempercepat analisis data perusahaan mereka. Dalam pelaksanaan pengolahan data, *entity matching* berperan mencocokkan dua entitas. Masalah dengan *entity matching* adalah bahwa kecocokan dalam pengenal unik terdistribusi jarang terjadi dan sering kali terkait dengan masalah privasi. Oleh karena itu diperlukan langkah untuk mencocokkan dua entitas yang sama yaitu dengan menggunakan *deep learning*. *Deepmatcher* adalah paket Python yang didasarkan pada arsitektur model *Deep learning* yang dianggap berfungsi baik dengan pencocokan entitas. Tujuan dari penelitian ini adalah untuk mengimplementasikan *deepmatcher* pada *entity matching* melalui pencocokan antara dua dataset *smartphone* dengan menggunakan model *SIF*, *RNN*, *attention*, dan *hybrid*. Hasil pengujian dengan semua model rata-rata akurat. Model *attention* dan *hybrid* cocok untuk proses pelatihan model pada dataset *smartphone* karena masing-masing memiliki nilai F1 terbesar yaitu 82,93.

Kata Kunci: dataset, data *smartphone*, *deep learning*, *deepmatcher*, *entity matching*

PENDAHULUAN

Perkembangan teknologi informasi dalam era modern saat ini membuat perusahaan atau instansi mulai berpikir bagaimana menciptakan keterbaruan teknologi yang dapat menguntungkan bisnis atau layanan yang diberikan. Banyak sekali penerapan teknologi terbaru dengan berbagai macam metode yang dilakukan telah berhasil dilakukan oleh berbagai perusahaan atau instansi. Salah satu contohnya adalah IBM mengembangkan IBM's Watson Chef initiative, yaitu sebuah teknologi digital yang memungkinkan penyaringan dan pemetaan lebih dari 10.000 resep untuk mempelajari bahan dan mengusulkan kombinasi resep baru dalam sebuah masakan (Magistretti et al., 2019). Penerapan perusahaan tersebut terhadap *Artificial Intelligence* (AI) dapat dikatakan bahwa prioritas utama perusahaan saat ini dalam penerapan teknologi informasi adalah kecepatan dan efektivitas dari *tools* yang digunakan atau yang dimanfaatkan. Selain itu, saat ini banyak perusahaan yang terus melakukan percepatan kemampuan analisis data. Contohnya adalah *the sensor messages* yang dihasilkan oleh otomatisasi manufaktur, dianggap sebagai salah satu jenis penerapan *big data* (Ko et al., 2019). Sehingga data juga menjadi bagian yang sangat penting untuk dilakukan percepatan.

Entity matching, yang mempunyai makna dalam menemukan contoh data yang merujuk ke entitas dunia nyata yang sama (Mudgal et al., 2018). Dalam penerapan ke pengolahan data, *Entity matching* berfungsi sebagai pencocokan dua entitas. Apabila terdapat dua sumber data berisi *shared unique identifier* misalnya nomor wajib pajak, masalah pencocokan entitas menjadi hal yang biasa seperti gabungan database (Brunner & Stockinger, 2019). Akan tetapi Brunner & Stockinger (2019) menambahkan bahwa *unique identifier* semacam itu jarang terjadi dan sering kali terkait dengan masalah privasi. Sehingga dibutuhkan sebuah metode atau langkah yang tepat untuk melakukan percobaan *entity matching,* dan *deep learning* merupakan salah satu contohnya.

Deep learning adalah sebuah paradigma yang sukses dalam bidang *machine learning* yang telah mencapai kesuksesan signifikan di berbagai bidang, seperti *computer vision*, *natural language processing*, *speech recognition*, *genomics*, dan lain-lain (Thirumuruganathan et al., 2020). Pendapat lain mengatakan bahwa *deep learning* merupakan sub bidang pembelajaran mesin yang mencoba mempelajari abstraksi tingkat tinggi dalam data dengan memanfaatkan arsitektur hierarkis (Guo et al., 2016). Sedangkan menurut Kamilaris & Prenafeta-Boldú (2018), *deep learning* termasuk dalam bidang komputasi pembelajaran mesin dan mirip dengan ANN (*Artificial Neural Networks*). Penelitian sebelumnya yang dilakukan oleh Mudgal et al. (2018) telah berhasil menerapkan *deep learning* dalam *entity matching. Entity matching* (juga dikenal sebagai resolusi entitas, deduplikasi, keterkaitan rekaman, pencocokan objek, pencocokan *fuzzy*, pemrosesan gabungan kesamaan, rekonsiliasi referensi) adalah bertugas untuk mengidentifikasi contoh objek atau entitas yang merujuk ke objek dunia nyata yang sama (Köpcke & Rahm, 2008). Dalam penerapan di banyak aplikasi, pengguna perlu memahami mengapa dua entitas dianggap cocok, yang mengungkapkan kebutuhan akan aturan *Entity Matching* yang dapat ditafsirkan dan ringkas (Singh et al., 2017).

Deepmatcher merupakan sebuah kerangka *entity matching* mendalam yang diusulkan oleh Mudgal et al. (2018), kerangka ini memiliki tiga modul yaitu penyematan atribut, representasi tujuan atribut, dan pengklasifikasi (Fu et al., 2020). Sementara itu menurut Chen et al. (2018), *deepmatcher* adalah sebuah paket Python berdasarkan arsitektur model *deep learning* yang dapat dengan baik menampilkan *entity matching, question answering,* dan *textual entailment*. Chen et al. (2018) menambahkan bahwa *deepmatcher* dibuat di atas PyTorch (Paszke et al., 2017), yaitu sebuah *framework* pembelajaran mendalam yang baik untuk penelitian karena memungkinkan iterasi pengembangan yang cepat. Melalui *deepmatcher* dapat diharapkan dengan baik dan cepat pencocokan dua buah entitas akan bisa dilakukan.

Dalam pengujian *deep learning* menggunakan 4 jenis model yaitu *SIF, RNN, Attention,* dan *Hybrid*. Model ini dikemukakan oleh Mudgal et al. (2018) yang kemudian secara ringkas dijelaskan oleh Chen et al. (2018) yaitu sebagai berikut:

1. *SIF*: Model ini pertama-tama mengumpulkan informasi tentang kata-kata yang ada di setiap contoh data, dan kemudian membandingkannya.
2. *RNN*: Model ini pertama-tama melakukan peringkasan peka urutan dari setiap instance data, lalu membandingkannya. Secara intuitif, contoh data yang berisi urutan kata yang mirip dianggap cocok.
3. *Attention*: Model ini pertama-tama melakukan penyelarasan antarkata di setiap contoh data, lalu melakukan perbandingan kata demi kata berdasarkan penyelarasan, dan terakhir menggabungkan informasi ini untuk melakukan klasifikasi. Secara intuitif, contoh data yang berisi kata-kata yang menyelaraskan dianggap cocok.

4. *Hybrid*: Model ini pertama-tama melakukan penyelarasan antara urutan kata di setiap contoh data, kemudian membandingkan urutan kata yang menyelaraskan, dan terakhir menggabungkan informasi ini untuk melakukan klasifikasi. Secara intuitif, contoh data yang berisi perataan urutan kata dianggap cocok.

Penelitian ini akan berfokus pada penerapan *deepmatcher* dalam *entity matching* melalui pencocokan antara dua buah *dataset*. *Dataset* yang digunakan adalah data spesifikasi ponsel dari GSMArena dan Amazon. Penerapan *deepmatcher* menggunakan *Jupyter Notebook*, yakni sebuah *notebook* komputasi *open source* populer yang memungkinkan penulis menggabungkan kode, visualisasi, dan teks dalam satu dokumen (file .ipynb) yang struktur dasarnya berupa JSON (Rule et al., 2018). Dalam melakukan penerapan *deepmatcher* akan digunakan beberapa algoritma perbandingan, yaitu *SIF*, *RNN*, *Attention*, dan *Hybrid*. Pengujian menggunakan keempat jenis algoritma ini akan menghasilkan seberapa akurat algoritma ini dalam penerapan *deppmatcher* terhadap *entity matching* di kedua *dataset* yang telah dipilih.

METODE PENELITIAN

Pengumpulan *Dataset*

Dataset yang digunakan dalam penelitian ini merupakan data spesifikasi *smartphone* yang berasal dari situs GSMArena dan Amazon. *Dataset* diperoleh dari situs web Kaggle, yang merupakan situs untuk berbagi data, catatan, diskusi, dan lain sebagainya. *Dataset* pada awalnya diperoleh sebanyak 10.680 data yang berasal dari situs GSMArena, serta sebanyak 4.176 data yang berasal dari situs Amazon (Hidayat et al., 2020). Contoh masing-masing *dataset* dapat dilihat pada Tabel 1 dan Tabel 2.

Tabel 1. Contoh *dataset* dari situs GSMArena

Oem	Display_size	Platform_os	Platform_chipset	Memory_internal
Acer	10.1 inches, 295.8 cm (~67.8% screen-to-body ratio)	Android 4.4 (KitKat)	Mediatek MT8127 (28 nm)	16GB 1GB RAM
Acer	3.8 inches, 41.1 cm (~55.0% screen-to-body ratio)	Microsoft Windows Mobile 6.5 Professional	Qualcomm QSD8250 Snapdragon S1	256MB RAM, 512MB ROM

Tabel 2. Contoh *dataset* dari situs Amazon

Product_name	By_info	Feature
Samsung Galaxy M10 (Ocean Blue, 3+32GB)	Samsung	13MP+5MP ultra-wide angle dual camera | 5MP f2.0 front camera. The internet usage time is 15 hours for 3G as well as 19 hours for LTE.The video playback time is 17 hours and audio playback time is 84 hours15.8cm (6.22") HD+ Infinity V Display with 90% screen ratio3GB RAM and 32GB internal memory face unlock | 3400 mAh lithium-ion batteryDual SIM (nano+nano) with dual standby and dual VoLTE1.6GHz Exynos 7870 octa-core processor | Android Oreo v8.1 OS

Samsung Galaxy M20 (Charcoal Black, 3+32GB)	Samsung	13MP+5MP ultra-wide dual camera | 8MP f2.0 front camera16cm (6.3") Full HD+ Infinity V Display with 2340x1080 crystal clear resolution (409 PPI)5000 mAh battery with 3x fast charge | 15W Type-C fast charger in the box3GB RAM and 32GB internal memory face unlock and fingerprint sensor | Dual SIM (nano+nano) with dual standby and dual VoLTEWidevine L1 certification for HD streaming | Dolby ATMOS 360 surround sound1.8GHz Exynos 7904 octa-core processor | Android Oreo v8.1 OS

Setelah data diperoleh maka masing-masing *dataset* dilakukan tahap *preprocessing* sebelum digunakan untuk pengujian. Selanjutnya *dataset* dieliminasi menggunakan fungsi *OverlapBlocker* yang terdapat pada *py_entitymatching package*. Proses eliminasi bertujuan untuk menghilangkan pasangan tupel yang tidak cocok serta data sampah yang tidak dibutuhkan dalam pengujian. Dari hasil proses eliminasi diperoleh *dataset* kandidat sebanyak 188 data yang akan digunakan dalam pengujian (Hidayat et al., 2020). Hasil *dataset* kandidat dapat dilihat pada Tabel 3.

Tabel 3. Hasil *dataset* kandidat

_id	**ltable_ id**	**rtable_ id**	**ltable_ Brand**	**ltable_Type**	**ltable_ Chipset**	**ltable_Screen_Dimension**
0	0	0	Samsung	Galaxy A10 (Black, 2GB RAM and 32GB) with No Cost EMI/Additional Exchange Offers	Exynos 7884 octa core processor	6.2-inch
1	2	1	Samsung	Galaxy A20 (Black, 3GB RAM, 32GB Storage) with No Cost EMI/Additional Exchange Offers	Exynos 7884 octa core processor	6.4-inch
			Samsung	Galaxy A10	Exynos 7884 Octa	6.2 inches, 95.9 cm (~81.6% screen-to-body ratio)
			Samsung	Galaxy A20	Exynos 7884	6.4 inches, 100.5 cm (~85.0% screen-to-body ratio)

Pemberian Label Pada Data

Setelah melakukan pengumpulan *dataset* maka selanjutnya adalah pemberian label ke setiap data untuk menentukan *match* dan *non-match* antar masing-masing *dataset*. Proses ini mengacu pada penjelasan (Jupyter, 2018). *Deepmatcher* perlu menampung dan memproses data terlebih dahulu sebelum dilakukan *training* data dan persiapan untuk dilakukan *neural network training*. Saat ini *deepmatcher* hanya mendukung file yang memiliki format CSV saja. Oleh karena itu setiap file CSV yang berisi data, harus memiliki jenis-jenis kolom sebagai berikut:

1. Atribut "*Left*": Bertujuan untuk mencocokkan pasangan tupel. Atribut "*Left*" adalah kolom yang sesuai dengan tupel "kiri" atau tupel pertama dalam pasangan tupel. Nama kolom ini diharapkan diawali dengan "left_" secara default.
2. Atribut "*Right*": Atribut "*Right*" adalah kolom yang sesuai dengan tupel "right" atau tupel kedua dalam pasangan tupel. Nama kolom ini diharapkan diawali dengan "right_" secara default.
3. Kolom label: Kolom yang berisi label untuk setiap pasangan tupel. Akan diberi nama "label" secara default.
4. Kolom ID: Kolom yang berisi ID unik untuk setiap pasangan tupel. Ini untuk kenyamanan evaluasi. Diharapkan diberi nama "id" secara default.

Hasil dari *dataset* yang telah diberi label dapat dilihat pada Tabel 4.

Tabel 4. Hasil pelabelan *dataset*

_id	**ltable_id**	**rtable_id**	**ltable_Brand**	**ltable_Type**	**ltable_Chipset**	**ltable_Screen_Dimension**	**Label**
0	0	0	Samsung	Galaxy A10 (Black, 2GB RAM and 32GB) with No Cost EMI/Additional Exchange Offers	Exynos 7884 octa core processor	6.2-inch	
1	2	1	Samsung	Galaxy A20 (Black, 3GB RAM, 32GB Storage) with No Cost EMI/Additional Exchange Offers	Exynos 7884 octa core processor	6.4-inch	
			Samsung	Galaxy A10	Exynos 7884 Octa	6.2 inches, 95.9 cm (~81.6% screen-to-body ratio)	1
			Samsung	Galaxy A20	Exynos 7884	6.4 inches, 100.5 cm (~85.0% screen-to-body ratio)	1

Dataset yang telah diberi label selanjutnya dibagi menjadi 3 jenis data, yaitu *train*, *valid*, dan *test*. Pembagian data ini menggunakan fungsi *data.split* dari *deepmatcher*. Rasio pembagian masing-masing jenis data adalah 3:1:1.

Model Neural Network dan Train Model

Pada tahapan ini dilakukan penggunaan jenis *neural network* yang dapat digunakan dalam *deepmatcher* untuk melakukan *entity matching.* Seperti yang dijelaskan pada jenis-jenis model yang digunakan pada *deep learning* yakni terdapat 4 jenis model yang dapat digunakan, di antaranya adalah *SIF, RNN, Attention,* dan *Hybrid.* Setiap model diuji dan dibandingkan dengan *dataset* dan konfigurasi parameter yang sama. Pada pengujian ini terlebih dahulu menentukan konfigurasi parameter awal yang optimal sesuai dengan

dataset yang dimiliki. Konfigurasi parameter yang digunakan yaitu nilai *epochs* sebesar 10, nilai *batch size* sebesar 16, serta nilai *positive to negative ratio* sebesar 3. Pengujian dilakukan menggunakan komputer dengan spesifikasi *processor* Intel Core i5-8265U 1.60GHz, memori RAM sebesar 8GB, serta sistem operasi Windows 10. Bahasa pemrograman Python yang digunakan merupakan versi 3.7. Kemudian menggunakan *buit-in network* pada *deepmatcher* dengan menggunakan *construct: model = dm.MatchingModel(attr_summarizer='<TYPE>').* Dengan *'<TYPE>'* berisi metode yang digunakan yaitu *SIF, RNN, Attention,* atau *Hybrid.*

Tahapan selanjutnya adalah melakukan *train model.* Pada tahapan ini melatih *neural network model* menggunakan data pelatihan dan validasi yang telah diproses. Terdapat beberapa parameter yang dapat digunakan untuk proses *train,* yaitu sebagai berikut:

1. *Train*: Objek set data pelatihan yang diproses (jenis *MatchingDataset*).
2. Validasi: Objek set data validasi yang diproses (dari jenis *MatchingDatase*t).
3. *Epochs*: Berapa kali seluruh data *train* untuk melatih model.
4. *Batch size*: Jumlah contoh berlabel (pasangan tupel) yang digunakan untuk setiap langkah pelatihan.
5. *Best save path*: Jalur untuk menyimpan model terbaik.
6. *Pos neg ratio*: Rasio bobot contoh positif (kecocokan) dengan bobot contoh negatif (bukan kecocokan).

Setelah *dataset* berhasil dikumpulkan dan diberi label, maka selanjutnya dilakukan pengujian serta proses *training* untuk masing-masing model yang ada pada *deepmatcher*. Langkah-langkah pengujian dan *training* untuk masing-masing model adalah sebagai berikut:

1. Pengujian dengan *Matching Model SIF* dan *Train Data*;
2. Pengujian dengan *Matching Mod*el *RNN* dan *Train Data*;
3. Pengujian dengan *Matching Model Attention* dan *Train Data*; dan
4. Pengujian dengan *Matching Model Hybrid* dan *Train Data*.

HASIL DAN PEMBAHASAN

Pengujian dengan *Matching Model SIF* dan *Train Data*

Pada pengujian dengan *matching* model *SIF* dilakukan percobaan sebanyak 10 kali. Setiap percobaan akan dilakukan proses *training model* dan *testing evaluation*. Pengujian ini akan menghasilkan nilai durasi proses, *precision*, *recall*, serta F1. Hasil proses *training model* dengan *matching model SIF* dapat dilihat pada Tabel 5. Sedangkan, hasil proses *testing evaluation* dengan *matching model SIF* dapat dilihat pada Tabel 6.

Tabel 5. Hasil proses *Training Model* dengan *Matching Model SIF*

Percobaan	Durasi Proses	Precision	Recall	F1
1	0.2	48.65	100.00	65.45
2	0.2	48.65	100.00	65.45
3	0.2	48.65	100.00	65.45
4	0.2	48.65	100.00	65.45
5	0.2	48.65	100.00	65.45
6	0.2	48.65	100.00	65.45
7	0.2	48.65	100.00	65.45

8	0.2	48.65	100.00	65.45
9	0.2	48.65	100.00	65.45
10	0.2	48.65	100.00	65.45

Tabel 6. Hasil proses *Testing Evaluation* dengan *Matching Model SIF*

Percobaan	**Durasi Proses**	**Precision**	**Recall**	**F1**
1	0.1	39.47	100.00	56.40
2	0.1	39.47	100.00	56.40
3	0.1	39.47	100.00	56.40
4	0.1	39.47	100.00	56.40
5	0.1	39.47	100.00	56.40
6	0.1	39.47	100.00	56.40
7	0.1	39.47	100.00	56.40
8	0.1	39.47	100.00	56.40
9	0.1	39.47	100.00	56.40
10	0.1	39.47	100.00	56.40

Berdasarkan hasil yang diperoleh pada Tabel 5 dan Tabel 6 dapat diketahui bahwa pada proses *training model* dan *testing evaluation* menghasilkan nilai durasi proses, *precision*, *recall*, serta F1 yang seluruhnya sama dan stabil. Dari tabel tersebut juga didapatkan nilai terbesar dari F1 pada proses *training model* sebesar 65,45 dan pada proses *testing evaluation* sebesar 56,60 yang dapat dikategorikan cukup akurat. Hasil ini diperoleh karena *matching model SIF* bekerja dengan mempertimbangkan kata-kata yang ada di setiap pasangan nilai atribut untuk menentukan *match* dan *non-match*, serta tidak memperhitungkan urutan kata (Mudgal et al., 2018). *Dataset* yang digunakan sendiri memiliki struktur kata yang hampir mirip di antara keduanya, serta jumlah data yang tidak begitu banyak.

Pengujian dengan *Matching Model RNN* dan *Train Data*

Pada pengujian dengan *matching model RNN* dilakukan percobaan sebanyak 10 kali. Setiap percobaan akan dilakukan proses *training model* dan *testing evaluation*. Pengujian ini akan menghasilkan nilai durasi proses, *precision*, *recall*, serta F1. Hasil proses *training model* dengan *matching model RNN* dapat dilihat pada Tabel 7. Sedangkan, hasil proses *testing evaluation* dengan *matching model RNN* dapat dilihat pada Tabel 8.

Tabel 7. Hasil proses *Training Model* dengan *Matching Model* RNN

Percobaan	**Durasi Proses**	**Precision**	**Recall**	**F1**
1	0.1	51.43	100.00	67.92
2	0.8	52.00	98.11	67.97
3	0.1	60.71	94.44	73.91
4	0.1	60.71	94.44	73.91
5	0.1	62.96	94.44	75.56
6	0.1	68.18	83.33	75.00
7	0.1	66.67	88.89	76.19
8	0.1	60.87	77.78	68.29
9	0.1	57.14	88.89	69.57

10	0.1	60.87	77.78	68.29

Tabel 8. Hasil proses *Testing Evaluation* dengan *Matching Model RNN*

Percobaan	Durasi Proses	Precision	Recall	F1
1	0.1	37.84	93.33	53.85
2	0.1	45.16	93.33	60.87
3	0.1	46.43	86.67	60.47
4	0.1	48.15	86.67	61.90
5	0.1	50.00	86.67	63.41
6	0.1	50.00	66.67	57.14
7	0.1	50.00	66.67	57.14
8	0.1	47.37	60.00	52.94
9	0.1	48.15	86.67	61.90
10	0.1	47.83	73.33	57.89

Berdasarkan hasil yang diperoleh pada Tabel 7 dan Tabel 8 dapat diketahui bahwa pada proses *training model* dan *testing evaluation* menghasilkan nilai durasi proses yang hampir seluruhnya sama sebesar 0,1, serta nilai *precision*, *recall*, dan F1 yang fluktuatif. Dari tabel tersebut juga didapatkan nilai terbesar dari F1 pada proses *training model* sebesar 76,19 dan pada proses *testing evaluation* sebesar 63,41 yang dapat dikategorikan cukup akurat. Hasil ini diperoleh karena *matching model RNN* bekerja dengan mempertimbangkan urutan kata yang ada di setiap pasangan nilai atribut untuk menentukan *match* dan *non-match* (Mudgal et al., 2018).

Pengujian dengan *Matching Model Attention* dan *Train Data*

Pada pengujian dengan *matching model Attention* dilakukan percobaan sebanyak 10 kali. Setiap percobaan akan dilakukan proses *training model* dan *testing evaluation*. Pengujian ini akan menghasilkan nilai durasi proses, *precision*, *recall*, serta F1. Hasil proses *training model* dengan *matching model Attention* dapat dilihat pada Tabel 9. Sedangkan, hasil proses *testing evaluation* dengan *matching model Attention* dapat dilihat pada Tabel 10.

Tabel 9. Hasil proses *Training Model* dengan *Matching Model Attention*

Percobaan	Durasi Proses	Precision	Recall	F1
1	0.3	56.25	100.00	72.00
2	0.3	64.29	100.00	78.26
3	0.4	64.29	100.00	78.26
4	0.6	73.91	94.44	82.93
5	0.9	68.00	94.44	79.07
6	0.6	73.91	94.44	82.93
7	0.6	73.91	94.44	82.93
8	0.6	73.91	94.44	82.93
9	0.6	73.91	94.44	82.93
10	0.6	72.73	88.89	80.00

Tabel 10. Hasil proses *Testing Evaluation* dengan *Matching Model Attention*

Percobaan	Durasi Proses	Precision	Recall	F1
1	0.2	38.24	86.67	53.06
2	0.2	54.55	80.00	64.86
3	0.2	54.55	80.00	64.86
4	0.4	61.11	73.33	66.67
5	0.4	57.14	80.00	66.67
6	0.4	57.89	73.33	64.71
7	0.4	57.89	73.33	64.71
8	0.4	57.89	73.33	64.71
9	0.4	57.89	73.33	64.71
10	0.4	57.89	73.33	64.71

Berdasarkan hasil yang diperoleh pada Tabel 9 dan Tabel 10 dapat diketahui bahwa pada proses *training model* diperoleh nilai durasi proses rata-rata di kisaran 0,6, serta pada proses *testing evaluation* diperoleh nilai durasi proses rata-rata di kisaran 0,4. Dari tabel tersebut juga didapatkan nilai *precision*, *recall*, dan F1 yang cenderung stabil mulai dari percobaan ke-6. Nilai F1 terbesar yang diperoleh pada proses *training model* sebesar 82,93 yang dapat dikategorikan akurat, serta pada proses *testing evaluation* sebesar 66,67 yang dapat dikategorikan cukup akurat. Hasil ini diperoleh karena *matching model Attention* bekerja dengan mempertimbangkan penyelarasan kata-kata yang ada di setiap pasangan nilai atribut untuk menentukan *match* dan *non-match*, serta tidak memperhitungkan urutan kata (Mudgal et al., 2018).

Pengujian dengan *Matching Model Hybrid* dan *Train Data*

Pada pengujian dengan *matching model Hybrid* dilakukan percobaan sebanyak 10 kali. Setiap percobaan akan dilakukan proses *training model* dan *testing evaluation*. Pengujian ini akan menghasilkan nilai durasi proses, *precision*, *recall*, serta F1. Hasil proses *training model* dengan *matching model Hybrid* dapat dilihat pada Tabel 11. Sedangkan, hasil proses *testing evaluation* dengan *matching model Hybrid* dapat dilihat pada Tabel 12.

Tabel 11. Hasil proses *Training Model* dengan *Matching Model Hybrid*

Percobaan	Durasi Proses	Precision	Recall	F1
1	0.4	64.29	100.00	78.26
2	0.5	60.00	100.00	75.00
3	0.5	65.38	94.44	77.27
4	0.5	70.83	94.44	80.95
5	0.5	70.83	94.44	80.95
6	0.4	70.83	94.44	80.95
7	0.4	70.83	94.44	80.95
8	0.4	73.91	94.44	82.93
9	0.4	70.83	94.44	80.95
10	0.4	76.19	96.23	81.6

Tabel 12. Hasil proses *Testing Evaluation* dengan *Matching Model Hybrid*

Percobaan	Durasi Proses	Precision	Recall	F1
1	0.3	52.63	66.67	58.82
2	0.3	45.83	73.33	56.41
3	0.4	55.00	73.33	62.86
4	0.3	58.82	66.67	62.50
5	0.3	63.16	80.00	70.59
6	0.3	60.00	80.00	68.57
7	0.3	63.16	80.00	70.59
8	0.3	63.16	80.00	70.59
9	0.3	60.00	80.00	68.57
10	0.3	63.16	80.00	70.59

Berdasarkan hasil yang diperoleh pada Tabel 11 dan Tabel 12 dapat diketahui bahwa dihasilkan nilai durasi proses yang hampir seluruhnya sama sebesar 0,4 pada proses *training model* dan sebesar 0,3 pada proses *testing evaluation*, serta nilai *precision*, *recall*, dan F1 yang cenderung fluktuatif. Dari tabel tersebut juga didapatkan nilai terbesar dari F1 pada proses *training model* sebesar 82,93 yang dapat dikategorikan akurat, serta pada proses *testing evaluation* sebesar 70,59 yang dapat dikategorikan cukup akurat. Hasil ini diperoleh karena *matching model Hybrid* bekerja dengan mempertimbangkan penyelarasan urutan kata yang ada di setiap pasangan nilai atribut untuk menentukan *match* dan *non-match* (Mudgal et al., 2018).

KESIMPULAN

Hasil pengujian menunjukkan bahwa *entity matching* menggunakan *deepmatcher* dapat digunakan pada *dataset smartphone* yang diperoleh dari situs GSMArena dan Amazon. Dari hasil pengujian dengan semua *matching model* yang disediakan oleh *deepmatcher* didapatkan hasil yang rata-rata cukup akurat. *Matching model Attention* dan *Hybrid* cocok digunakan untuk proses *training model* pada *dataset smartphone* karena masing-masing memiliki nilai F1 terbesar yaitu 82,93. Serta, *matching model Hybrid* cocok digunakan untuk proses *testing evaluation* pada *dataset smartphone* karena memiliki nilai F1 terbesar yaitu 70,59. Hasil yang didapatkan dirasa kurang maksimal karena *dataset* yang digunakan sendiri memiliki struktur kata yang hampir mirip di antara keduanya, serta jumlah data yang tidak begitu banyak.

DAFTAR PUSTAKA

Brunner, U., & Stockinger, K. (2019). Entity matching on unstructured data: an active learning approach. *2019 6th Swiss Conference on Data Science (SDS)*, 97–102.

Chen, C., Halevy, A., & Tan, W. (2018). BigGorilla : An Open-Source Ecosystem for Data Preparation and Integration. *Data Engineering Bulletin*, 10–22.

Fu, C., Han, X., He, J., & Sun, L. (2020). Hierarchical matching network for heterogeneous entity resolution. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 3665–3671.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187, 27–48.

Hidayat, R., Pratama, R. P., & Rakhmawati, N. A. (2020). *Smartphone Datasets for Research*. Doi. 10.5281/zenodo.4284357.

Jupyter. (2018). Getting started with DeepMatcher. https://nbviewer.jupyter.org/github/anhaidgroup/deepmatcher/blob/master/examples/getting_started.ipynb.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and*

Electronics in Agriculture, 147, 70–90.

Ko, Y.-C., Ting, Y.-Y., & Fujita, H. (2019). A visual analytics with evidential inference for big data: case study of chemical vapor deposition in solar company. *Granular Computing*, 4(3), 531–544.

Köpcke, H., & Rahm, E. (2008). Training selection for tuning entity matching. *QDB/MUD*, 3–12.

Magistretti, S., Dell'Era, C., & Messeni Petruzzelli, A. (2019). How intelligent is Watson? Enabling digital transformation through artificial intelligence. *Business Horizons*, 62(6), 819–829. Doi. 10.1016/j.bushor.2019.08.004.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., & Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. *Proceedings of the 2018 International Conference on Management of Data*, 19–34.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in Pytorch. *NIPS 2017 Autodiff Workshop: The Future of Gradient-Based Machine Learning Software and Techniques*.

Rule, A., Tabard, A., & Hollan, J. D. (2018). Exploration and explanation in computational notebooks. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–12.

Singh, R., Meduri, V., Elmagarmid, A., Madden, S., Papotti, P., Quiané-Ruiz, J.-A., Solar-Lezama, A., & Tang, N. (2017). Generating concise entity matching rules. *Proceedings of the 2017 ACM International Conference on Management of Data*, 1635–1638.

Thirumuruganathan, S., Tang, N., Ouzzani, M., & Doan, A. (2020). Data Curation with Deep Learning. *EDBT*, 277–286.