JISKa, Vol. 6, No. 3, SEPTEMBER, 2021, Pp. 130 – 138
ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)

Implementasi Deep Learning untuk Entity Matching pada *Dataset* Obat (Studi Kasus K24 dan Farmaku)

Rivanda Putra Pratama [(1)], Rahmat Hidayat [(2)*], Nisrina Fadhilah Fano [(3)] , Adam Akbar [(4)] , Nur Aini Rakhmawati [(5)]
Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember, Surabaya
e-mail : {rivanda.19052,rahmat.19052}@mhs.its.ac.id, {adamakbar.id,nisrina.fano}@gmail.com, nur.aini@is.its.ac.id.
* Penulis korespondensi.
Artikel ini diajukan 18 Januari 2021, direvisi 6 Maret 2021, diterima 13 Maret 2021, dan dipublikasikan 22 September 2021.

Abstract

Data processing speed in companies is important to speed up their analysis. Entity matching is a computational process that companies can perform in data processing. In conducting data processing, entity matching plays a role in determining two different data but referring to the same entity. Entity matching problems arise when the dataset used in the comparison is large. The deep learning concept is one of the solutions in dealing with entity matching problems. DeepMatcher is a python package based on a deep learning model architecture that can solve entity matching problems. The purpose of this study was to determine the matching between the two datasets with the application of DeepMatcher in entity matching using drug data from farmaku.com and k24klik.com. The comparison model used is the Hybrid model. Based on the test results, the Hybrid model produces accurate numbers, so that the entity matching used in this study runs well. The best accuracy value of the 10th training with an F1 value of 30.30, a precision value of 17.86, and a recall value of 100.

Keywords: Entity Matching, Deep Learning, DeepMatcher, Dataset, Hybrid

Abstrak

Kecepatan dalam pengolahan data di perusahaan penting dilakukan untuk mempercepat analisis perusahaan. *Entity matching* menjadi sebuah proses komputasi yang dapat dilakukan oleh perusahaan dalam pengolahan data. Dalam melakukan pemrosesan data, *entity matching* berperan dalam menentukan dua data berbeda namun merujuk ke entitas yang sama. Permasalahan dalam *entity matching* muncul ketika *dataset* yang digunakan dalam perbandingan berukuran besar. Konsep *deep learning* menjadi salah satu solusi dalam menangani permasalahan *entity matching. DeepMatcher* adalah sebuah paket Python berdasarkan arsitektur model *deep learning* yang dapat menjadi solusi dalam permasalahan *entity matching.* Tujuan dari penelitian ini adalah untuk mengetahui pencocokan antara dua buah *dataset* dengan penerapan *DeepMatcher* dalam *entity matching* dengan menggunakan data obat yang berasal dari farmaku.com dan k24klik.com. Model perbandingan yang digunakan adalah model *Hybrid*. Berdasarkan hasil pengujian dari model *Hybrid* menghasilkan angka yang akurat, sehingga *entity matching* yang digunakan pada penelitian ini berjalan dengan baik. Dari hasil *training*, diperoleh nilai akurasi terbaik pada *training* ke-10 dengan nilai *F1* sebesar 30,30, *precision* sebesar 17,86, dan *recall* sebesar 100.

Kata Kunci: ***Entity Matching*, *Deep Learning*, *DeepMatcher*, *Dataset*, *Hybrid***

1. PENDAHULUAN

Perkembangan dan kecepatan pengolahan data di suatu perusahaan saat ini sangat penting untuk dilakukan, langkah-langkah tersebut dilakukan untuk mempercepat analisis mengenai kemampuan perusahaan dalam membaca kebutuhan konsumen. *Entity matching* menjadi sebuah proses komputasi yang dapat dilakukan oleh perusahaan dalam pengolahan data. *Entity matching* merupakan permasalahan untuk menentukan apakah dua data merujuk ke entitas yang sama di dunia nyata (Mudgal et al., 2018). Contoh kasus dalam *entity matching* yang biasanya terjadi adalah menentukan kecocokan data dari dua *dataset* berbeda, padahal merujuk ke suatu

entitas yang sama (Christophides et al., 2015). Permasalahan muncul ketika *dataset* yang digunakan berukuran besar. Jika *dataset* yang digunakan besar, akan cukup sulit untuk menentukan pasangan entri yang cocok (Li et al., 2020). Proses yang dilakukan untuk menyelesaikan permasalahan *entity matching* menjadi kompleks. Beberapa peneliti telah menemukan metode untuk menyelesaikan permasalahan *entity matching* pada *dataset* yang berukuran besar. Salah satunya adalah menggunakan konsep *deep learning*.

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). Selain itu, *deep learning* telah berhasil diterapkan diberbagai bidang seperti *signal processing, artificial intelligence*, dan *emotion detection* (Abdullah et al., 2021). *Deep learning* menjadi sebuah *tool* yang secara mendalam dan berukuran besar yang berfokus pada jaringan saraf tiruan atau *artificial neural networks* (Yuan et al., 2020). *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., 2019). Dalam penelitian (Chen et al., 2018), mengungkapkan bahwa *DeepMatcher* adalah sebuah paket python berdasarkan arsitektur model *deep learning* yang dapat dengan baik menampilkan *entity matching, question answering*, dan *textual entailment*. Kemudian Mudgal et al. (2018) berpendapat bahwa *DeepMatcher Python* dapat menjadi sebuah solusi dalam permasalahan *entity matching*.

Penelitian yang dilakukan oleh Li et al. (2020) telah melakukan *deep entity matching* dengan model bahasa yang sudah dilakukan *pre-training* sebelumnya (*Pre-Trained Models*). Melalui sistem yang bernama DITTO, para peneliti melakukan *entity matching* dalam tiga tahap, yaitu tahapan *blocking*, tahap pelatihan sistem, dan tahap *matching*. Pada tahap *blocking,* dilakukan pemangkasan dari pasangan entitas-entitas. Pada tahap kedua, proses pelatihan dilakukan dengan menambahkan *domain knowledge* untuk meningkatkan performa DITTO. Pada tahap ketiga, proses *matching* dilakukan untuk menemukan pasangan entitas yang tepat (Li et al., 2020). Penelitian lain yang menggunakan *Deep Learning* untuk *Entity Matching* adalah penelitian yang dilakukan oleh Fu et al. (2019). Pada penelitian tersebut, para peneliti mengembangkan model *neural multi-perspective matching* (MPM). Terdapat empat *layer* pada model MPM. *Layer* pertama adalah *attribute representation layer* yang berfungsi untuk menyeragamkan bentuk atribut yang ada pada data. *Layer* kedua adalah *comparison layer* yang berfungsi untuk membandingkan nilai dari atribut menggunakan nilai *similarity* antara dua nilai atribut. *Layer* ketiga adalah *selection layer* yang berfungsi untuk memilih data yang mempunyai keserupaan antar atribut yang paling tinggi. Kemudian data tersebut digabungkan pada *layer* keempat, yaitu *aggregation layer* untuk mendapatkan hasil dari *entity.*

Pengembangan sebuah sistem untuk *entity matching* yang terdiri dari enam tahap telah dilakukan oleh Kasai et al. (2019). Tahap pertama adalah *input representation.* Pada tahap tersebut, dilakukan tokenisasi untuk nilai atribut dan melakukan vektorisasi kata untuk mendapatkan representasi masukan. Tahap kedua adalah *attribute representations* yang menggunakan *universal bidirectional RNN* untuk mendapatkan vektor atribut menggunakan representasi masukan dari setiap nilai atribut. Selanjutnya adalah *attribute similarity* yang berfungsi untuk menghitung keserupaan setiap pasang entitas pada data. Hasil perhitungan keserupaan atribut kemudian digunakan pada tahap *record similarity* untuk menghitung keserupaan antar entitas. Tahap kelima adalah *matching classification* yang berfungsi untuk memisahkan pasangan entitas yang *match* dengan pasangan entitas yang tidak *match.* Tahap terakhir pada sistem ini adalah *training objectives* yang bertujuan untuk melatih sistem.

Penelitian yang dilakukan oleh Zhao & He (2019) dalam mengembangkan sebuah sistem *end-to-end auto* EM yang terdiri dari tiga komponen. Komponen pertama adalah *attribute-type detection*, yakni pada komponen ini dilakukan identifikasi tipe atribut. Sebagai contoh apabila salah satu atribut berisi tentang nama orang, maka atribut tersebut akan ditandai sebagai tipe 'Person'. Model identifikasi tipe atribut telah dilatih sebelumnya menggunakan data *Knowledge Base* dari mesin pencarian. Komponen kedua adalah *attribute-level EM model,* di mana pada

komponen ini dilakukan penghitungan keserupaan antara nilai dua atribut. Model ini dibagi menjadi dua bagian, yaitu *type-specific models,* yaitu untuk atribut dengan tipe yang sudah terdeteksi dan *unified model* untuk atribut yang tipenya masih belum terdeteksi. Komponen ketiga adalah *table-level EM* yang menghasilkan hasil akhir dari solusi untuk permasalahan *entity matching* (Zhao & He, 2019). Penelitian dalam menganalisa *entity matching* dengan menggunakan model SIF, RNN, Attention, dan Hybrid telah dilakukan oleh (Hidayat et al., 2021). Dalam penelitian tersebut melakukan pencocokan pada *dataset smartphone* dengan menggunakan *DeepMatcher.* Hasilnya adalah 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.

Pengujian dalam *deep learning* dapat menggunakan 4 jenis model yaitu SIF, RNN, *Attention*, dan *Hybrid*. Model ini dikemukakan oleh Mudgal et al. (2018) yang selanjutnya diringkas oleh Chen et al. (2018) sebagai berikut.

- SIF: Model ini berperan dalam mengumpulkan informasi tentang kata-kata yang ada di setiap contoh data, dan kemudian membandingkannya.
- RNN: Model ini melakukan peringkasan dari urutan setiap *instance data*, kemudian dilakukan perbandingan. Secara intuitif, contoh data yang berisi urutan kata yang mirip dianggap cocok.
- *Attention*: Model ini melakukan penyelarasan antar kata 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 selaras dianggap cocok.
- *Hybrid*: Model ini 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.

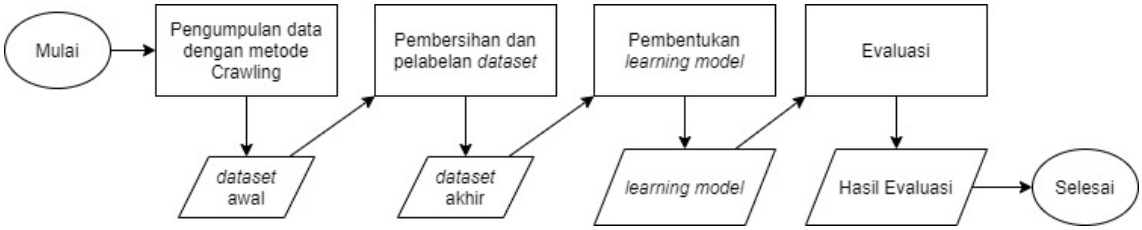
Penelitian ini bertujuan untuk mengetahui pencocokan antara dua buah *dataset* dengan penerapan *DeepMatcher* dalam *entity matching*. *Dataset* yang digunakan adalah data obat yang berasal dari farmaku.com dan k24klik.com. Dalam penerapan *DeepMatcher* menggunakan Jupyter Notebook, yaitu sebuah *notebook* komputasi *open source* populer yang memungkinkan pengguna dapat menggabungkan kode, visualisasi, dan teks dalam satu dokumen (file.ipynb) yang memiliki struktur dasarnya berupa JSON (Rule et al., 2018). Pada penelitian ini akan menggunakan salah satu model perbandingan, yaitu *Hybrid*. Pengujian menggunakan model ini akan menghasilkan seberapa akurat model ini dalam penerapan *DeepMatcher* terhadap *entity matching* di data obat dari farmaku.com dan k24klik.com.

2. METODE PENELITIAN

Penelitian ini memiliki 4 proses utama yang dilakukan secara berurutan yaitu *data crawling,* pembersihan dan pelabelan data (*data cleansing and labeling)*, pembentukan *learning model*, dan evaluasi. Gambar 1 menunjukkan proses metode penelitian dalam penelitian ini, pada masing-masing proses memiliki *output* yang diperlukan untuk menjalankan proses selanjutnya. Penjelasan yang lebih lengkap mengenai proses-proses tersebut akan dipaparkan pada bab ini.

2.1. *Crawling*

Dalam melakukan pengumpulan data, pada penelitian ini menggunakan tehnik *crawling*. *Crawling* merupakan sebuah metode dalam melakukan pengumpulan data pada sebuah situs web dengan memasukkan URL (*Uniform Resource Locator*) yang menjadi acuan dalam melakukan pencarian (Arsi et al., 2021). *Dataset* yang digunakan pada penelitian ini dihimpun dengan teknik *web crawling* dari dua situs yang menjual obat secara *online* di Indonesia yaitu: farmaku.com sebagai S1, dan k24klik.com sebagai S2 (Akbar et al., 2021). Data obat yang dikumpulkan dibatasi hanya pada obat dengan kategori yang sama pada kedua situs yaitu “Batuk dan Pilek” dan “Obat Generik”. Proses *crawling* dilakukan pada bulan Januari, 2021. Hasil *crawling* tersebut disimpan ke dalam *file* berekstensi *comma-separated values* (CSV) untuk masing-masing sumber.



Gambar 1. Diagram Alir Metode Penelitian.

2.2. Pembersihan dan Pelabelan *Dataset*

Dataset menggunakan ekstensi CSV yang berupa data obat yang berhasil dihimpun kemudian dilakukan pembersihan dengan menghapus tanda koma (,) yang terdapat pada setiap baris agar tidak terjadi ambigu untuk nilai kolom. Kemudian, dilakukan juga penghapusan pada kolom-kolom yang tidak beririsan antar kedua sumber untuk dapat membandingkan kedua entitas secara setara. Sebagai contoh, seperti yang dapat dilihat pada Gambar 2, terdapat dua tabel yang memuat data obat dengan kolom yang berbeda, jika dilakukan penghapusan pada kolom yang tidak beririsan maka kolom "Dikirim Dari" pada Gambar 2A dan kolom "Diproduksi Oleh" pada Gambar 2B akan dihapus.

Nama	Dikirim Dari	Komposisi	Khasiat
Obat A	Jakarta	Parasetamol	Meredakan batuk
Obat B	Jakarta	Parasetamol	Meredakan demam
Obat C	Jakarta	Vitamin C	Meningkatkan imun

A

Nama	Komposisi	Manfaat	Diproduksi Oleh
Obat A	Parasetamol	Meredakan batuk	PT. XYZ
Obat B	Parasetamol	Meredakan demam	PT. XYZ
Obat C	Vitamin C	Meningkatkan imun	PT. XYZ

B

Gambar 2. Contoh Data Obat.

Setelah dilakukan penghapusan kolom, langkah selanjutnya adalah menyatukan data dari kedua sumber dengan cara memasangkan setiap baris data pada S1 dengan setiap baris data pada S2 untuk mencapai semua probabilitas yang ada. Sebagai contoh, jika kita memasangkan data pada Gambar 2A dan Gambar 2B maka hasil akan terlihat seperti Tabel 1.

Tabel 1. Contoh Hasil Kombinasi Antar Sumber.

No	Data A				Data B			
	Nama	Dikirim Dari	Komposisi	Khasiat	Nama	Komposisi	Manfaat	Diproduksi Oleh
1	Obat A	Jakarta	Parasetamol	Meredakan batuk	Obat A	Parasetamol	Meredakan batuk	PT. XYZ
2	Obat B	Jakarta	Parasetamol	Meredakan demam	Obat A	Parasetamol	Meredakan batuk	PT. XYZ
3	Obat C	Jakarta	Vitamin C	Meningkatkan imun	Obat A	Parasetamol	Meredakan batuk	PT. XYZ
4	Obat A	Jakarta	Parasetamol	Meredakan batuk	Obat B	Parasetamol	Meredakan demam	PT. XYZ
5	Obat B	Jakarta	Parasetamol	Meredakan demam	Obat B	Parasetamol	Meredakan demam	PT. XYZ
6	Obat C	Jakarta	Vitamin C	Meningkatkan imun	Obat B	Parasetamol	Meredakan demam	PT. XYZ
7	Obat A	Jakarta	Parasetamol	Meredakan batuk	Obat C	Vitamin C	Meningkatkan imun	PT. XYZ
8	Obat B	Jakarta	Parasetamol	Meredakan demam	Obat C	Vitamin C	Meningkatkan imun	PT. XYZ
9	Obat C	Jakarta	Vitamin C	Meningkatkan imun	Obat C	Vitamin C	Meningkatkan imun	PT. XYZ

Selanjutnya, pasangan data yang benar-benar berbeda akan dieliminasi (*blocking*) untuk mengurangi *learning* yang tidak diperlukan oleh *DeepMatcher*. Eliminasi baris pasangan data

pada penelitian ini dilakukan dengan menggunakan *package* Python dari AnHai's Group juga yang bernama py_entitymatching. Pada py_entitymatching terdapat pengaturan *overleap* yang bernilai desimal, yang dimaksudkan untuk jumlah kata yang sama pada kedua atribut untuk kriteria data yang dipertahankan. Sebagai contoh jika nilai *overleap* adalah 2 dan atribut yang dibandingkan adalah "Khasiat" dan "Manfaat, maka data nomor 1, 5, dan 9 pada Tabel 1 akan dipertahankan dan data nomor 2, 3, 4, 6, 7, dan 8 akan dihapus karena jumlah kata yang sama tidak mencapai dua. Nilai *overleap* bersifat minimal dalam artian jika jumlah kata yang sama melebihi nilai *overleap* maka baris tersebut tidak dieliminasi.

Baris-baris pasangan data yang tidak tereliminasi pada langkah sebelumnya akan menjadi *dataset* kandidat yang kemudian akan dipecah menjadi tiga *dataset* yang diperlukan oleh *DeepMatcher* yaitu *training* dan *validation* untuk pembentukan model dan *testing* untuk pengukuran akurasi model, pemecahan *dataset* kandidat dilakukan secara acak dengan rasio 3:1:1 sesuai dengan rekomendasikan AnHai Doan (Mudgal, et al., 2018) di mana 3 untuk *dataset training* dan 1 untuk *dataset validation* dan *testing*.

Setiap baris pasangan data pada tiga *dataset* tersebut akan dilabeli secara manual dengan nilai 0 untuk pasangan data yang tidak serupa atau 1 untuk pasangan data yang serupa untuk digunakan oleh *DeepMatcher* sebagai acuan dalam pembentukan *learning model*.

2.3. *Learning Model*

DeepMatcher menyediakan empat jenis *deep learning* (Mudgal et al., 2018) yaitu Smooth Inverse Frequency (*sif*) yang mempertimbangkan kecocokan antar entitas berdasarkan nilai atribut tanpa memperhitungkan urutannya, Bidirectional RNN (*rnn*) yang mempertimbangkan urutan nilai atribut antar entitas, Decomposable Attention Model (*att*) yang mempertimbangkan keselarasan nilai setiap atribut antar entitas, dan Hybrid Model (*hybrid*) yang mempertimbangkan keselarasan urutan nilai atribut antar entitas.

Selain *dataset*, ada beberapa nilai yang perlu ditetapkan pada *DeepMatcher* untuk melakukan pemodelan, nilai tersebut adalah *epochs*, *batch_size*, dan *pos_neg_ratio* yang semuanya ditetapkan dalam bentuk bilangan desimal. *Epoch* dimaksudkan untuk jumlah iterasi latihan yang dilakukan oleh *DeepMatcher*, *batch_size* mengacu pada jumlah baris data yang akan dipelajari oleh *DeepMatcher* untuk setiap langkah pada satu latihan (*training step*), dan *pos_neg_ratio* adalah nilai bobot untuk *dataset training* yang ditentukan berdasarkan rasio jumlah baris berlabel 0 dan 1 pada *dataset* tersebut sebagai penyeimbang dalam pembentukan *learning model*. Semua parameter yang sudah ditentukan tadi akan mempengaruhi kinerja dan akurasi dari *learning model*.

Pemodelan pada tahap latihan didasarkan pada *dataset training* dengan kata lain *DeepMatcher* 'mempelajari' pelabelan yang dilakukan oleh penulis berdasarkan atribut yang terdapat pada *dataset* tersebut. *Dataset validation* digunakan untuk mengevaluasi hasil latihan sebagai acuan untuk menentukan *learning model* terbaik berdasarkan skor F1 dari *dataset validation* tersebut. *DeepMatcher* akan menyimpan *learning model* yang terbaik dari semua pelatihan yang telah dilakukan untuk nanti digunakan dalam melakukan prediksi kecocokan.

2.4. Evaluasi

$$prec = \frac{TP}{(TP+FP)} \qquad (1)$$

$$rec = \frac{TP}{(TP+FN)} \qquad (2)$$

$$F1 = \frac{2 \times prec \times recall}{(prec+recall)} \qquad (3)$$

Akurasi dari *learning model* dievaluasi dengan pengukuran *precision* (*prec*), *recall* (*rec*), dan *F1-score* (*f-measure*) yang merupakan pengukuran populer dalam *machine learning* (Garreta &

Moncecchi, 2013). *Precision* atau ketepatan berkaitan dengan kemampuan sistem untuk tidak memanggil dokumen yang tidak relevan (Hardi, 2006). Secara sederhana *prec* dapat diartikan sebagai perbandingan antara prediksi positif yang tepat (TP)(*true positive*) dan semua prediksi positif yang dihasilkan mesin (TP + FP)(*false positive*). Persamaan untuk nilai *prec* dapat dilihat pada persamaan (1). Sementara itu, *recall* merupakan sebuah cara evaluasi yang sering dilakukan dalam mengukur hasil eksperimen dalam *machine learning* (Powers, 2020). Dalam persamaan *rec* dapat diartikan sebagai perbandingan antara prediksi positif yang tepat (TP) dan semua baris yang sebenarnya positif (TP + FN) (*false negative*). Persamaan untuk nilai *rec* dapat dilihat pada persamaan (2). Skor F1 dapat diartikan sebagai nilai *mean* dari *prec* dan *rec* [2], dengan persamaan yang dapat dilihat pada persamaan (3). Berdasarkan pengukuran tersebut maka dapat disimpulkan bahwa *learning model* yang paling akurat dalam artian dapat memprediksi kecocokan tanpa kesalahan adalah *learning model* yang memiliki nilai 1 untuk skor F1-nya atau 100 jika disajikan dalam bentuk persen.

3. HASIL DAN PEMBAHASAN

3.1. *Dataset*

Crawling pada S1 didapatkan 198 baris untuk total dengan perincian 114 baris untuk kategori "Batuk Pilek" dan 84 baris untuk kategori "Obat Generik", dan atribut yang didapatkan pada S1 adalah nama produk, URL produk, harga produk, satuan produk, deskripsi produk, indikasi penggunaan, kategori produk, komposisi produk, dan dosis penggunaan produk. *Crawling* pada S2 didapatkan 243 baris dengan perincian 97 baris untuk kategori "Batuk Pilek" dan 146 baris untuk kategori "Obat Generik", dan atribut yang didapatkan pada S2 adalah nama produk, URL produk, harga produk, satuan produk, deskripsi produk, indikasi penggunaan, kategori produk, *tag*, komposisi produk, dan dosis penggunaan produk. Untuk rangkuman hasil *crawling* dapat dilihat pada Tabel 2.

Tabel 2. Rangkuman Hasil *Crawling*.

	S1	**S2**
Jumlah baris kategori Batuk Pilek	114	97
Jumlah baris kategori Obat Generik	84	146
Total baris	198	243
Atribut	nama produk URL produk harga produk satuan produk deskripsi produk indikasi penggunaan kategori produk komposisi produk dosis penggunaan produk	nama produk URL produk harga produk satuan produk deskripsi produk indikasi penggunaan kategori produk *tag* komposisi produk dosis penggunaan produk

Setelah itu, semua data yang berhasil dihimpun dilakukan pembersihan dengan menghapus semua tanda koma (,) pada semua nilai seperti yang sudah disebutkan pada sub-bab **Error! Reference source not found.**. Kemudian penghapusan kolom yang tidak beririsan, pada hasil *crawling* atribut *tag* pada S2 dihapus karena tidak beririsan dengan S1. Setelah itu dilakukan kombinasi data antara kedua sumber yang menghasilkan $198 \times 243 = 48.114$ baris yang kemudian dilakukan eliminasi pada baris yang benar-benar tidak mirip. Eliminasi dilakukan dalam dua tahap, tahap pertama, eliminasi dilakukan dengan membandingkan nilai atribut nama produk pada hasil kombinasi sebelumnya dengan nilai *overleap* adalah 2 yang menyisakan 566 baris pasangan data. Tahap kedua dilakukan dengan membandingkan nilai atribut satuan produk pada

hasil eliminasi tahap pertama dengan nilai *overleap* adalah 1 yang menyisakan 436 baris pasangan data. Hasil eliminasi tersebut menjadi *dataset* kandidat yang kemudian dipecah menjadi 3 *dataset* dengan rasio 3:1:1 sehingga hasil akhir pembentukan *dataset* menjadi 262 baris untuk *dataset training*, dan 87 baris untuk masing-masing *dataset validation* dan *testing*.

3.2. Inisialisasi DeepMatcher

Dalam menjalankan algoritma *DeepMatcher*, perlu adanya inisialisasi awal paket *DeepMatcher*. Inisialisasi awal dilakukan dengan mengunduh paket *DeepMatcher* ke modul Python. Setelah paket berhasil diunduh dan terinstal pada modul Python, maka selanjutnya adalah inisialisasi *library DeepMatcher* dengan memanggil *DeepMatcher*, *Torch*, dan OS.

3.3. Inisialisasi Data *Training*, *Validation*, *Testing*

Pada penelitian ini, *dataset* obat yang berasal dari situs K24klik dan Farmaku telah digabung menjadi satu dokumen berformat CSV. Proses penggabungan *dataset* mengikuti standar yang telah dibuat oleh algoritma *DeepMatcher*, yaitu menggabungkan kedua *dataset* menjadi *left table* (*ltable*) dan *right table* (*rtable*). Setelah dilakukan penggabungan *dataset*, maka selanjutnya adalah proses pelabelan. Proses pelabelan dilakukan secara manual ke seluruh *dataset*. Label pada algoritma *DeepMatcher* terbagi menjadi 2, yaitu *match* yang bernilai 1 dan *non-match* yang bernilai 0. Proses berikutnya adalah membagi *dataset* menjadi 3 data, yaitu *training, validation,* dan *testing.* Perbandingan antara data *training, validation,* dan *testing* adalah 3:1:1, sehingga didapatkan data *training* sejumlah 262, data *validation* sejumlah 87, serta data *testing* sejumlah 87. Data *training, validation,* dan *testing* yang digunakan pada penelitian ini dapat dilihat pada (Akbar et al., 2021). Proses yang terakhir adalah inisialisasi data *training*, *validation,* dan *testing* menggunakan fungsi *data.process* yang dimiliki oleh *library DeepMatcher*. Proses inisialisasi ini berfungsi untuk mengenali *dataset* yang digunakan, serta membersihkan *dataset* sesuai kebutuhan sistem.

3.4. Proses *Training Dataset*

Data yang digunakan dalam proses *training* adalah data *training* dan *validation*. Proses *training* pada penelitian ini menggunakan model *Hybrid* yang terdapat pada algoritma *DeepMatcher*. Konfigurasi proses *training* pada penelitian ini menggunakan nilai *epoch* sebesar 10 dan *batch_size* sebesar 16. Konfigurasi ini dipilih karena mengingat keterbatasan sumber daya dan spesifikasi *hardware* yang digunakan. Serta, nilai *pos_neg_ratio* sebesar 17, yang didapatkan dengan membagi banyaknya label *non-match* dengan label *match* pada data *training* dan *validation*.

Proses *training dataset* pada penelitian ini dilakukan sebanyak 10 kali sesuai dengan nilai *epoch* yang diberikan. Proses *training data* menghasilkan nilai akurasi terbaik yang ditunjukkan dengan nilai *F1* sebesar 30,30, *precision* sebesar 17,86, dan *recall* sebesar 100. Dari hasil akurasi yang diperoleh, terlihat bahwa nilai *Precision* rendah. Hal ini dimungkinkan karena perbandingan label *match* dan *non-match* yang tidak seimbang, dimana data berlabel *match* jauh lebih sedikit dibandingkan dengan data berlabel *non-match*.

3.5. Prediksi Kecocokan Data

Dalam mendapatkan prediksi kecocokan data, perlu membuat sebuah *dataset* yang berisi data obat yang berasal dari kedua situs. *Dataset* ini juga dibuat dengan format yang disesuaikan dengan standar algoritma *DeepMatcher* tetapi tanpa menggunakan kolom Label. *Dataset* ini biasa disebut dengan data *unlabeled*. Data *unlabeled* yang digunakan pada penelitian ini berjumlah 10 dan dapat dilihat pada (Akbar et al., 2021). Data *unlabeled* ini juga perlu dilakukan proses inisialisasi dengan fungsi *data.process_unlabeled* yang dimiliki oleh *library DeepMatcher*.

Setelah data *unlabeled* berhasil di inisialisasi, maka proses selanjutnya adalah melakukan proses prediksi kecocokan data menggunakan fungsi *model.run_prediction*. Proses prediksi kecocokan data ini akan menghasilkan *match_score* berdasarkan urutan pada data *unlabeled. Match_score* ini yang dapat digunakan sebagai acuan apakah kedua data cocok atau tidak. Pada penelitian

ini, kedua data akan dikatakan cocok apabila *match_score* lebih dari 0,9 karena dianggap paling mendekati nilai 1. Proses yang terakhir adalah menyimpan hasil prediksi kecocokan data ke dalam file dokumen berformat CSV dengan fungsi predictions.to_csv. Hasil prediksi kecocokan data secara lengkap dapat dilihat pada (Akbar et al., 2021).

Dari 10 data yang digunakan pada proses prediksi kecocokan data, terdapat 4 data yang dapat dikatakan cocok karena *match_score* yang diperoleh lebih dari 0,9. Selanjutnya, keempat data ini digunakan sebagai data perbandingan harga obat yang dapat dilihat pada **Error! Reference source not found.**.

No.	Farmaku		K24klik		*Match Score*
	Nama Obat	**Harga**	**Nama Obat**	**Harga**	
1	Decadryl Exp 120 ml	Rp 17.700	Decadryl Exp Syr 120ml	Rp 21.357	0,937635779
2	Mucera Drop	Rp 48.800	Mucera Drop 15mg/1ml 15ml	Rp 54.078	0,907262385
3	OBH Combi Flu Jahe 60 ml New	Rp 13.500	Obh Combi Dewasa Batuk Flu Jahe 60ml	Rp 13.867	0,977404475
4	Ciprofloxacin 500 mg Tab Hexp	Rp 600	Ciprofloxacin Hexpharm 500mg Tab 50s	Rp 1.326	0,92949605

Berdasarkan data perbandingan obat, dapat diketahui bahwa untuk produk obat pertama yang bernama Decadryl Exp 120 ml pada situs Farmaku memiliki harga lebih murah dibandingkan K24klik dengan harga Rp 17.700. Produk obat kedua bernama Mucera Drop pada situs Farmaku lebih murah dibandingkan K24klik dengan harga Rp 48.800. Produk obat ketiga bernama OBH Combi Flu Jahe 60ml pada situs Farmaku lebih murah dibandingkan K24klik dengan harga Rp 13.500. Serta, produk obat keempat bernama Ciprofloxacin Hexpharm 500 mg pada situs Farmaku lebih murah dibandingkan K24klik dengan harga Rp 600.

4. KESIMPULAN

Penelitian ini mengimplementasikan *deep learning* untuk *entity matching* pada *dataset* obat yang diperoleh dari situs K24Klik dan Farmaku. Penelitian ini menggunakan *library DeepMatcher* untuk mengolah *dataset*. Proses *training* pada penelitian ini dilakukan sebanyak 10 kali menggunakan model *Hybrid*. Dari hasil *training*, diperoleh nilai akurasi terbaik yang ditunjukkan dengan nilai *F1* sebesar 30,30, *precision* sebesar 17,86, dan *recall* sebesar 100. Nilai *Precision* yang rendah dimungkinkan karena data berlabel *match* jauh lebih sedikit dibandingkan dengan data berlabel *non-match*. Namun, percobaan prediksi terhadap 10 data menunjukkan hasil yang sesuai. Sehingga, metode *entity matching* yang digunakan pada penelitian ini dapat dikatakan cukup baik dan dapat digunakan untuk kebutuhan lanjutan seperti membandingkan harga pada produk obat yang sama.

DAFTAR PUSTAKA

Abdullah, S. M. S. A., Ameen, S. Y. A., M. Sadeeq, M. A., & Zeebaree, S. (2021). Multimodal Emotion Recognition using Deep Learning. *Journal of Applied Science and Technology Trends*, *2*(02), 52–58. https://doi.org/10.38094/jastt20291

Akbar, A., Fano, N. F., Pratama, R. P., Hidayat, R., & Rakhmawati, N. A. (2021). *Dataset Obat Untuk Penelitian Entity Matching*. https://doi.org/10.5281/ZENODO.4445466

Arsi, P., Wahyudi, R., & Waluyo, R. (2021). Optimasi SVM Berbasis PSO pada Analisis Sentimen Wacana Pindah Ibu Kota Indonesia. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(2), 231–237. https://doi.org/10.29207/resti.v5i2.2698

Chen, C., Golshan, B., Halevy, A., Tan, W., & Doan, A. (2018). BigGorilla: An Open-Source Ecosystem for Data Preparation and Integration. *IEEE Data Eng. Bull.*, *41*, 10–22.

Christophides, V., Efthymiou, V., & Stefanidis, K. (2015). Entity Resolution in the Web of Data. *Synthesis Lectures on the Semantic Web: Theory and Technology*, *5*(3), 1–122. https://doi.org/10.2200/S00655ED1V01Y201507WBE013

Fu, C., Han, X., Sun, L., Chen, B., Zhang, W., Wu, S., & Kong, H. (2019). End-to-End Multi-Perspective Matching for Entity Resolution. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 4961–4967. https://doi.org/10.24963/ijcai.2019/689

Garreta, R., & Moncecchi, G. (2013). *Learning Scikit-Learn: Machine Learning in Python*. Packt Publishing Ltd.

Hardi, W. (2006). Mengukur kinerja search engine : sebuah eksperimentasi penilaian precision and recall untuk informasi ilmiah bidang ilmu perpustakaan dan informasi [Search Engines performance evaluation: an experimental the value of precision and recall for scientific information in LIS field.]. In *Visi Pustaka [National Library of Indonesia]*. Perpustakaan Nasional RI [National Library of Indonesia].

Hidayat, R., Pratama, R. P., & Rakhmawati, N. A. (2021). ANALISIS ENTITY MATCHING PADA DATASET SMARTPHONE MENGGUNAKAN METODE SIF, RNN, ATTENTION, DAN HYBRID. *TEKNOSAINS: MEDIA INFORMASI SAINS DAN TEKNOLOGI*, *15*(1), 67–77. https://doi.org/10.24252/teknosains.v15i1.17583

Kasai, J., Qian, K., Gurajada, S., Li, Y., & Popa, L. (2019). Low-resource Deep Entity Resolution with Transfer and Active Learning. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5851–5861.

Li, Y., Li, J., Suhara, Y., Doan, A., & Tan, W.-C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, *14*(1), 50–60. https://doi.org/10.14778/3421424.3421431

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., & Raghavendra, V. (2018). Deep Learning for Entity Matching. *Proceedings of the 2018 International Conference on Management of Data*, 19–34. https://doi.org/10.1145/3183713.3196926

Powers, D. M. W. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, *2*(1), 37–63.

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. https://doi.org/10.1145/3173574.3173606

Thirumuruganathan, S., Tang, N., Ouzzani, M., & Doan, A. (2020). Data Curation with Deep Learning. *Proceedings of the 23rd International Conference on Extending Database Technology (EDBT)*, 277–286. https://doi.org/https://dx.doi.org/10.5441/002/edbt.2020.25

Yuan, Q., Shen, H., Li, T., Li, Z., Li, S., Jiang, Y., Xu, H., Tan, W., Yang, Q., Wang, J., Gao, J., & Zhang, L. (2020). Deep learning in environmental remote sensing: Achievements and challenges. *Remote Sensing of Environment*, *241*, 111716. https://doi.org/10.1016/j.rse.2020.111716

Zhao, C., & He, Y. (2019). Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. *The World Wide Web Conference on - WWW '19*, 2413–2424. https://doi.org/10.1145/3308558.3313578