e-ISSN 2549-7472
Edumatic: Jurnal Pendidikan Informatika
Vol. 6 No. 1, Juni, 2022, **Hal. 58-67**
DOI: 10.29408/edumatic.v6i1.5274

Klasifikasi Teks menggunakan Genetic Programming dengan Implementasi Web Scraping dan Map Reduce

Wirarama Wedashwara[1,*], Budi Irmawati [1], Andy Hidayat Jatmika [1], Ariyan Zubaidi [1]

[1] Program Studi Teknik Informatika, Universitas Mataram, Indonesia
[*] Correspondence: wirarama@unram.ac.id

Received: 7 Maret 2022 | Revised: 30 Maret 2022 | Accepted: 5 April 2022 | Published: 20 Juni 2022

Abstrak
Klasifikasi dokumen text pada media *online* menjadi permasalahan data besar dan memerlukan otomatisasi. Penelitian telah mengembangkan sistem klasifikasi teks dengan *pre-processing* menggunakan *map-reduce* dan pengumpulan data menggunakan *web scraping*. Penelitian ini bertujuan untuk melakukan evaluasi kinerja klasifikasi teks dengan menggabungkan algoritma genetik programming, map reduce dan *web scraping* untuk pemrosesan data besar berbentuk teks. Pengumpulan data dilakukan dengan observasi berbasis web scraping telah dikumpulkan data dengan mengurangi duplikat sebanyak 8126. *Map-reduce* telah melakukan tokenisasi dan *stop-word removal* dengan total 28507 term dengan 4306 *term* unik dan 24201 term duplikasi. Evaluasi klasifikasi teks menunjukkan single tree menghasilkan akurasi lebih baik (0,7072) dari decision tree (0,6874) dan terendah adalah *multi tree* (0,6726). Untuk perolehan nilai *support genetic programming* dengan *multi tree* menghasilkan rata-rata support tertinggi yaitu 0.3854 diikuti decision tree dengan 0,3584 dan paling kecil single tree dengan 0,3494. Secara umum jumlah support tidak sejalan dengan nilai akurasi yang dicapai.

Kata kunci: genetic programming; klasifikasi teks; map reduce

Abstract
Classification of text documents on online media is a big data problem and requires automation. Research has developed a text classification system with pre-processing using map-reduce and web scraping data collection. This study aims to evaluate text classification performance by combining genetic programming algorithms, map-reduce and web scraping for processing large data in the form of text. Data collection was carried out by observing web-based scraping. Data was collected by reducing 8126 duplicates. Map-reduce has tokenized and stopped-word removal with 28507 terms with 4306 unique terms and 24201 duplication terms. Text classification evaluation shows that a single tree produces better accuracy (0.7072) than a decision tree (0.6874), and the lowest is a multi-tree (0.6726). For the acquisition of genetic programming support values with the multi-tree, the highest average support is 0.3854, followed by the decision tree with 0.3584 and the smallest single tree with 0.3494. In general, the amount of support is not in line with the accuracy value achieved.

Keywords: *genetic programming; klasifikasi teks; map reduce*

PENDAHULUAN

Klasifikasi dokumen text pada media online menjadi permasalahan data besar dan memerlukan otomatisasi (Pintye et al., 2021). Akurasi klasifikasi teks dapat menurun jika terdapat banyak term yang ambigu antar *class* (Altınel & Ganiz, 2018). Melakukan pengelompokkan term untuk data yang besar memerlukan pemrosesan paralel (Du & Li, 2019). *Hadoop Map Reduce* merupakan *framework* pemrosesan parallel untuk data besar yang sudah

banyak digunakan sebagai platform OLAP (*Online Analytic Processing)* (Jeong & Cha, 2019). *Hadoop Map Reduce* juga sudah banyak digunakan untuk pemrosesan text pada data besar (Ranjitha et al., 2020).

Penelitian mempresentasikan klasifikasi teks menggunakan *genetic programming* dengan melakukan *pre-processing text* menggunakan *hadoop map reduce* dan pengumpulan data menggunakan *web scraping* (Tahmassebi & Gandomi, 2018). *Genetic programming* digunakan karena melakukan *association rule mining* (ARM) sebelum klasifikasi teks sehingga mendapatkan analisis tambahan untuk pola data besar (Thomas & Mathur, 2019). Data yang digunakan artikel dari *science-direct* dengan kata kunci *Internet of Things*, *big data* dan *machine learning* (Telikani et al., 2020).

Penelitian terkait klasifikasi teks menggunakan algoritma berbasis tree telah dilakukan menggunakan *decision tree* sebagai *feature selection* (Deng et al., 2019), *term weighting scheme* untuk *short*-klasifikasi teks (Alsmadi & Hoon, 2019) dan klasifikasi teks dan *clustering* dari *Twitter* data untuk *business analytics* (Halibas et al., 2018). Ketiga penelitian tersebut menggunakan *decision tree* yaitu algoritma yang akan dibandingkan dengan genetic *programming*. Selain itu belum ada penelitian yang menggabungkan dengan *pre-proccessing text* menggunakan *map-reduce*.

Selain itu, penelitian mengenai pemanfaatan map reduce untuk pre-processing sudah dilakukan meninjau aspek algoritmic dari pemrosesan parallelnya (Koutris et al., 2018), *Scalable Distributed Data Processing* (Anjum, 2018), hingga *Effective processing* untuk *unstructured* data menggunakan python (Kousalya & Parvez, 2018). Penelitian yang diusulkan menggunakan bahasa pemrograman python dan parallel processing. Tetapi menggunakan jenis *pre-processing* dan algoritma yang berbeda. Penelitian melakukan klasifikasi teks secara umum dan tidak melakukan sentimen analisis seperti penelitian yang sudah ada (Sihombing et al., 2021).

Pemanfaatan *genetic programming* untuk keperluan pemrosesan text sudah pernah dilakukan Automated selection dan configuration dari *multi-label grammar based* (de Sá et al., 2018) dan *feature selection* pada *highly dimensional skewed* data (Viegas et al., 2018). Kedua penelitian tidak melibatkan *web scraping* dan *map reduce* seperti pada penelitian ini. Penelitian ini juga menggunakan perbandingan *single tree* dan *multi tree* model dalam melakukan *rule extraction*.

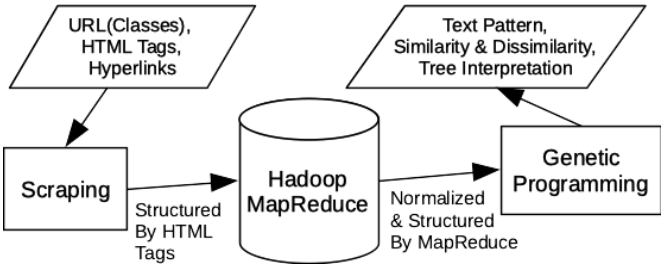
Berdasarkan hal tersebut, penelitian ini bertujuan untuk melakukan klasifikasi teks dengan analisis pola data berbasis ARM. Penelitian juga bertujuan untuk melakukan evaluasi kinerja klasifikasi teks dengan menggabungkan algoritma genetik programming, *map reduce* dan *web scraping* untuk pemrosesan data besar berbentuk teks. Melalui ARM diharapkan dapat diketahui pola data antar label yang dapat berpengaruh pada perolehan akurasi. Penelitian ini juga bertujuan membentuk sistem dari pengumpulan data melalui *web scraping*, pre-processing menggunakan *hadoop map-reduce* dan klasifikasi teks menggunakan *genetic programming*.

Evaluasi diawali dengan pembahasan data yang telah dikumpulkan menggunakan *web scraping* dan *map reduce* hingga penjabaran tokenisasi kata (Ramsingh & Bhuvaneswari, 2018). Selanjutnya dilakukan perbandingan antara model *single tree* dan *multi tree* pada *genetic programming*. Terakhir dilakukan perbandingan hasil akurasi dengan algoritma *decision tree* yang dianggap memiliki kesamaan sifat.

METODE

Data dikumpulkan melalui metode observasi terhadap website science direct dari 1 januari tahun 2020 hingga 1 desember 2021 menggunakan web scraping. Gambar 1 menunjukkan gambaran umum sistem. Data dikumpulkan melalui proses web scraping menggunakan library scrapy pada bahasa pemrograman python. Proses web scraping dilakukan melalui mengekstrak teks dari tag html yang spesifik dari halaman HTML sumber yaitu

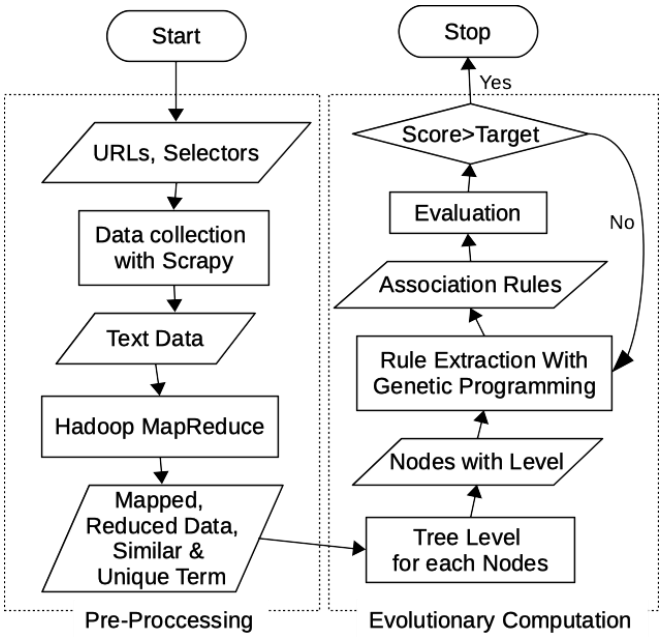
science-direct. Label class dipisahkan berdasarkan kata kunci pencarian pada form pencarian science direct yaitu kata kunci *Internet of Things*, *Big Data* dan *Machine Learning*.



Gambar 1. Gambaran umum sistem

Penyimpanan dilakukan pada *hadoop* untuk dilakukan proses *map reduce*. Proses *map reduce* memungkinkan pemrosesan secara paralel sehingga cocok untuk pemrosesan data yang banyak. Proses map dilakukan untuk memisahkan kata pada artikel yang di kumpulkan. Proses *stopword removal* juga dilakukan pada proses *map*. Pada proses *reduce* dilakukan proses *tokenization* yaitu menghitung jumlah kemunculan kata. Perhitungan *tokenization* pada *reduce* dilakukan dengan memisah kemunculan kata yang hanya terjadi di dalam satu label maupun muncul secara umum.

Genetic programming digunakan untuk melakukan ektraksi *text pattern*, perhitungan *similarity* dan *dissimilarity*, *tree interpretation* hingga tujuan utama penelitian yaitu klasifikasi teks. Data yang diproses oleh *genetic programming* adalah data yang sudah dalam bentuk objek yang dibuat melalui proses *map reduce* pada *hadoop*.

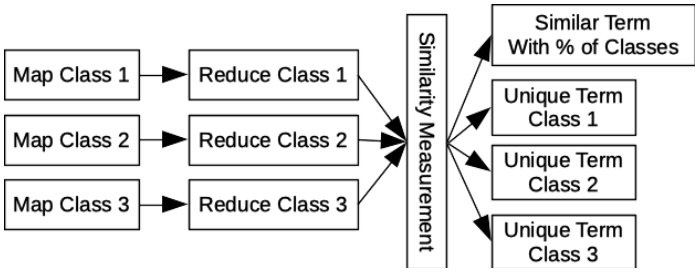


Gambar 2. Diagram alir sistem

Gambar 2 menunjukkan *flowchart* dari sistem. Proses terbagi menjadi *pre-proccessing* dan *evolutionary computation*. Proses *pre-proccessing* terdiri dari input URL (*science direct*) dan tag yang akan di download oleh *scrapy*. *Text* data diproses oleh *map reduce* hingga menjadi objek yang siap diproses oleh *genetic programming*.

Genetic programming melakukan tingkatan kata berdasarkan frekuensi kemunculannya. *Genetic programming* akan melakukan ekstraksi *rule* dengan memprioritaskan kata dengan

frekuensi kemunculan tinggi ke rendah. *Rule* yang diekstraksi akan digunakan untuk melakukan klasifikasi hingga mencapai target akurasi yang diharapkan.



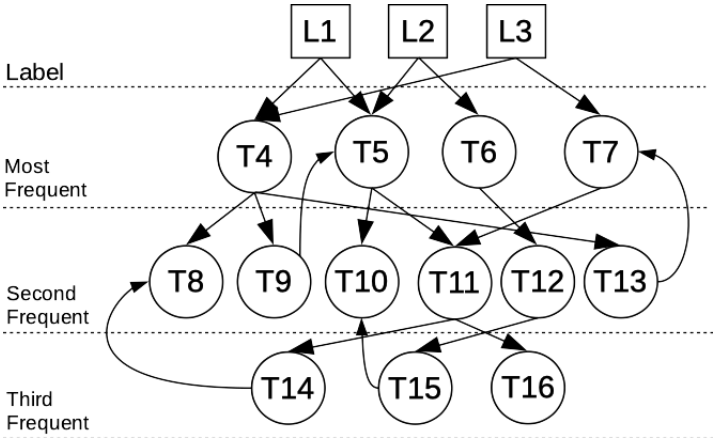
Gambar 3. Struktur *map reduce*

Gambar 3 menunjukkan proses *map reduce* pada sistem. Proses *map* dilakukan berdasarkan data per *class* dengan memisahkan *term* (kata) dari artikel yang di download. Proses *reduce* juga dilakukan berdasarkan masing-masing *class* dengan melakukan *tokenization* untuk *term* yang sudah di *map* sebelumnya. Proses akhir dilakukan dengan memisahkan kata-kata yang hanya muncul di *class* masing-masing sebagai Unik *term* dan *similar term*. Unik *term* akan digunakan untuk membentuk *specific rules*, sedangkan *similar term* digunakan untuk membentuk *common rule* pada *genetic programming*.

HASIL DAN PEMBAHASAN

Hasil

Pada bagian ini dijelaskan *struktur gen rule extractor* dari *genetic programming* model *single tree* dan *multi tree* yang digunakan sebagai *rule based classifier* pada penelitian ini. Pembahasan mencakup *struktur gen* dalam tampilan *grafik tree*, struktur objek dalam pemrograman dan *contoh rules* yang dihasilkan oleh masing-masing *rule extractor*. *Rule* yang dihasilkan oleh *rule extractor* hanya ditampilkan sebagian karena keterbatasan halaman.



Gambar 4. Struktur gen pada *single tree*

Gambar struktur *gen genetic programming* dengan model *single tree* ditunjukkan oleh gambar 4. Pada struktur *single tree* seluruh node tergabung dalam satu pohon dengan *root* lebih dari satu yang berisikan label atau kata kunci dalam pencarian. Node dengan bentuk persegi berisi label. Node dengan bentuk lingkaran berisi *term* yaitu kata dan frekuensi kemunculannya. Tingkatan pohon terbagi menjadi 4 bagian yaitu label dan 3 tingkatan frekuensi kemunculan kata yang direpresentasikan oleh masing-masing node.

Tabel 1. Struktur gen pada *single tree*

i	**NT_i**	**Lv_i**	**C_i**	**T_i**
1	L	0	4,5	IoT
2	L	0	5,6	BD
3	L	0	4,7	ML
4	T	1	8,9,13	Internet
5	T	1	10,11	*Algorithm*
6	T	1	12	*Data*
7	T	1	11	*Initialization*
8	T	2	-	*Nodes*
9	T	2	5	*Sensor*
10	T	2	-	*Storage*
11	T	2	14,16	*Cloud*
12	T	2	15	*Tree*
13	T	2	7	*Learning*
14	T	3	8	*Knowledge*
15	T	3	10	*Classification*
16	T	3	-	*Clustering*

Struktur *tree* pada *single tree* merupakan *directed graph* yang memungkinkan *multiple direction* dari satu node ke node lainnya. Direction juga tidak selalu dari atas kebawah tetapi juga memungkinkan naik ke node yang ada pada tingkatan diatasnya. Sehingga walaupun single tree memungkinkan ekstaksi *rule* yang sangat bervariatif. Tabel 1 menunjukkan struktur gen dari *genetic programming* dengan *rule extractor single-tree* pada gambar 4. Kolom pertama i menunjukkan indeks dari node 1 hingga 16. Kolom NT_i menunjukkan node type dari masing-masing node. Tipe L menunjukkan label yaitu *Internet of Things* (IoT), *Big Data* (BD) dan *Machine Learning* (ML). Sedangkan tipe T menunjukkan *term* yaitu kata yang muncul dalam artikel yang sudah dikumpulkan melalui proses web scraping. Kolom Lv_i menunjukkan tingkatan pada struktur pohon yaitu 0 untuk *root*, 1 untuk *most frequent*, 2 untuk *second* dan 3 untuk *third most frequent*. Kolom C_i menunjukkan koneksi antar node. Untuk *multiple connection* direpresentasikan dengan array pada pemrograman. T_i menunjukkan *term* yaitu kata yang direpresentasikan oleh masing-masing node.

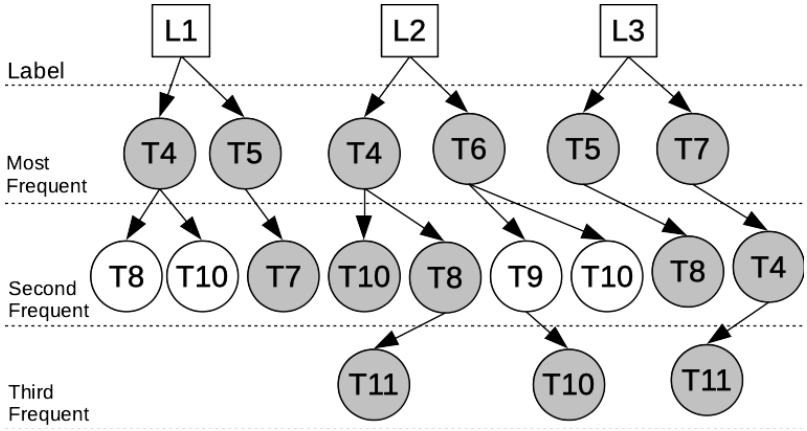
Tabel 2 menunjukkan rule yang diekstraksi oleh *single tree*. Tanda panah pada struktur *rules* pada kolom pertama menunjukkan pemisah antara *precedent* dan *dependent*. *Precedent* ditempatkan oleh label dan *dependent* menunjukkan *frequent item set*. *Length* menunjukkan jumlah node pada dependent. *Confidence* dan support menunjukkan hasil evaluasi *association rule mining* yang dapat dilihat pada penelitian sebelumnya. *Score* berisi gabungan *length*, *confidence* dan *support* yang ditunjukkan pada tabel 2. Hanya *rule* oleh L1 saja yang ditunjukkan oleh tabel 2 karena keterbatasan halaman. Rule yang diekstraksi bisa lebih banyak yang berasal dari L2 dan L3. *Rule* yang di ekstraksi berupa *incremental* dan tidak selalu harus berakhir hingga paling bawah. Semakin pendek rule bisa menghasilkan support yang lebih tinggi karena syarat yang lebih sedikit. Tetapi manjadi kurang kuat untuk digunakan pada pemrosesan klasifikasi atau regresi. Sehingga dipriotitaskan pada panjang rule untuk menghasilkan syarat yang lebih spesifik untuk menentukan label.

Struktur *multi tree* pada *genetic programming* ditunjukkan oleh gambar 5. Berbeda dengan *single tree* antar label tidak ada node yang terhubung. Sehingga terdapat term yang duplikat antar tree seperti T4 yang terdapat pada setiap *tree*. Berbeda dengan struktur *single tree* yang memungkinkan balik ke node diatasnya, dalam satu *tree* juga terdapat *term* yang

sama seperti T10 yang terdapat 3 duplikat pada tree L2. Pada representasi grafik multi *tree* tampak lebih sederhana tetapi lebih rumit pada struktur objek yang akan dibawas selanjutnya.

Tabel 2. Rule yang diektraksi oleh *single tree*

Rules	*Length*	*Confidence*	*Support*	*Score*
$L1 \rightarrow T4 \wedge T13$	2	0,3023	0,204	2,355
$L1 \rightarrow T4 \wedge T13 \wedge T7$	3	0,239	0,354	3,473
$L1 \rightarrow T4 \wedge T13 \wedge T7 \wedge T11$	4	0,219	0,189	4,298
$L1 \rightarrow T4 \wedge T13 \wedge T7 \wedge T11 \wedge T14$	5	0,1612	0,0712	5,151
$L1 \rightarrow T4 \wedge T13 \wedge T7 \wedge T11 \wedge T14 \wedge T8$	6	0,014	0,013	6,021
Rata-rata				4,259



Gambar 5. Struktur gen *multiple tree*

Struktur objek pada multi *tree* ditunjukkan pada tabel 3. Kolom i, NT_i, Lv_i, C_i dan V_i memiliki fungsi yang sama dengan single tree. Kolom L_i memiliki fungsi untuk menunjukkan kepemilikan oleh label. Jumlah L_i dan Lv_i selalu sama untuk menunjukkan tingkatan node dalam masing-masing *tree*. Karena tidak adanya direksi untuk balik ke node diatasnya sehingga Lv_i juga bisa di duplikasi pada satu *tree* yang sama seperti T10.

Tabel 3. Struktur gen multi *tree*

i	**NT_i**	**L_i**	**Lv_i**	***C_i***	***V_i***
1	L		0	[4,5]	IoT
2	L		0	[4,6]	BD
3	L		0	[5,7]	ML
4	T	1,2,3	1,1,2	[8,10],[10,8],[11]	Internet
5	T	1,3	1	[7],[7,8]	Algorithm
6	T	2	1	[9,10]	Data
7	T	1,3	1,2	[],[14]	Initialization
8	T	1,2,3	2,2,2	[],[11],[]	Nodes
9	T	2	2	[10]	Tree
10	T	1,2,2,2	2,2,2,3	[],[],[],[]	Storage
11	T	2,3	3,3	[],[]	Cloud

Connection Ci ditunjukkan dengan *array* tambahan. Jumlah *array* selalu sama dengan jumlah Li dan Lvi. Jika tidak ada koneksi selanjutnya akan berisi *array* kosong. Struktur objek ini memungkinkan representasi node tanpa melakukan duplikasi pada anggota *array*.

Tabel 4. *Rule* yang diekstraksi multi *tree*

Rules	*Length*	*Conf*	*Support*	*Score*
$L1 \rightarrow T4 \wedge T8$	2	0,234	0,241	2,358
$L1 \rightarrow T4 \wedge T10$	2	0,213	0,376	2,482
$L1 \rightarrow T5 \wedge T7$	2	0,242	0,241	2,362
$L2 \rightarrow T4 \wedge T10$	2	0,181	0,423	2,513
$L2 \rightarrow T4 \wedge T10$	2	0,325	0,172	2,334
$L2 \rightarrow T4 \wedge T8$	2	0,462	0,536	2,767
$L2 \rightarrow T4 \wedge T8 \wedge T10$	3	0,381	0,442	3,632
Rata-rata				2,635

Tabel 4 menunjukkan contoh *rule* yang diekstrak oleh struktur multi *tree*. Contoh menunjukkan sebagian *rule* yang diekstraksi oleh L1 dan L2. L1 menunjukkan maksimal dua kombinasi dependent karena hanya terdapat dua tingkat. Sedangkan L2 bisa mencapai tiga kombinasi karena memiliki tiga tingkatan. Perbedaan hasil dengan *single tree* secara spesifik akan dibahas pada sub bab selanjutnya.

Pembahasan

Tabel 5 menunjukkan data yang sudah dikumpulkan melalui proses web scraping. Data dikumpulkan dari artikel terbaru per 1 desember 2021 hingga batas 3000 judul untuk masing-masing kata kunci IoT, *Big Data* (BD) dan *Machine Learning* (ML). *Header horizontal* menunjukkan masing-masing kata kunci yang scraping dan totalnya. Masing-masing kata kunci dikoleksi sebanyak 3000 judul dan abstrak dan total berjumlah 9000.

Tabel 5. Data yang telah dikumpulkan melalui *web scraping*

	IoT	**BD**	**ML**	**Total**
Collected	3000	3000	3000	9000
Used	2561	2837	2728	8126
Duplikasi	191	258	425	874
IoT	0	65	126	191
BD	98	0	160	258
ML	125	300	0	425

Header vertikal menunjukkan keterangan jumlah yang dikumpulkan, dipakai, total duplikat yang sama pada setiap kata kunci dan relasi yang menunjukkan dengan kata kunci apa terjadi duplikat. Sebagai contoh label IoT memiliki total 191 duplikat yaitu 65 dengan BD dan 126 dengan ML. Duplikat terbanyak dimiliki oleh BD sebanyak 258 yaitu 98 dengan IoT dan 160 dengan ML. Dengan mengurangi total 874 artikel maka total data yang dipakai adalah 8126. Melalui pengumpulan duplikat ini bisa diperkirakan false positive yang akan muncul antar label karena persamaannya pada hasil pencarian.

Hasil *map reduce* pada tabel 6 menunjukkan kata yang diekstraksi dari artikel yang telah dikumpulkan sebelumnya melalui proses web scraping. Pada proses *map* dilakukan pemisahan setiap kata pada artikel. Jumlah kata yang ditunjukkan oleh tabel adalah kata-kata spesifik yang telah melalui proses *stopword removal* sebelumnya. Pada proses *map* diekstrak 28507 kata dari seluruh kata kunci.

Pada proses *reduce* dilakukan proses *tokenizer* dengan menghitung kata yang sama untuk mendapatkan term frequency nya. Selanjutnya dihitung kata yang hanya muncul pada masing-masing label (Unik) dan juga terdapat di pada kata kunci lain (Duplikasi) atau yang juga disebut *inverse document frequency*. Tiga baris bawah menunjukkan kesamaan kata kata kunci dengan

yang lainnya. Misalnya kata kunci IoT memiliki total 6203 kesamaan kata yaitu 2718 dengan BD dan 3485 dengan ML. Total kata yang sama berjumlah 28507 dan hanya 4306 kata unik yang akan digunakan yang akan digunakan untuk membuat pohon *rule extractor* pada *genetic programming*. Kata unik terbanyak dimiliki oleh IoT yaitu 1982 diikuti ML yaitu 1201 dan BD yaitu 1123.

Tabel 6. Pengumpulan kata menggunakan *map reduce*

	IoT	**BD**	**ML**	**Total**
Map	45378	43278	52012	140668
Reduce	9832	8392	10283	28507
Unik	1982	1123	1201	4306
Duplikasi	6203	7930	10068	24201
IoT	0	2718	3485	6203
BD	2983	0	4947	7930
ML	5269	4799	0	10068

Perbandingan akurasi *text classiofication* dengan *decision tree* pada tabel 7 menunjukkan perbandingan akurasi klasifikasi teks antara *genetic programming* dengan *decision tree*. Evaluasi mencakup *support* dari *rules* yang dihasilkan dan akurasi hasil klasifikasi teks. Evaluasi dilakukan dengan data mulai dari 1626 hingga seluruh data yaitu 8126.

Tabel 7. Perbandingan akurasi klasifikasi teks dengan *decision tree*

Data	***GP Single Tree***		***GP Multi Tree***		***Decision Tree***	
	Support	**Akurasi**	***Support***	**Akurasi**	***Support***	**Akurasi**
1626	0,172	0,601	0,262	0,542	0,242	0,667
3251	0,264	0,681	0,288	0,634	0,332	0,665
4876	0,298	0,712	0,343	0,643	0,332	0,612
6501	0,441	0,741	0,511	0,762	0,432	0,772
8126	0,572	0,801	0,523	0,782	0,454	0,721
Rata-rata	0,349	0,7072	0,385	0,6726	0,358	0,687

Untuk data yang paling sedikit yaitu 1626, *decision tree* memiliki hasil akurasi yang paling tinggi yaitu 0,667. Hasil ini menunjukkan *decision tree* memiliki akurasi yang lebih baik dengan data *training* yang lebih sedikit. *Single tree* memiliki hasil akurasi yang lebih tinggi dibanding *decision tree* sejak jumlah data 3251. *Multi tree* hanya menghasilkan akurasi lebih baik dari *decision tree* pada jumlah data 4876 dan 8126 saja. Secara rata-rata *genetic programming* dengan *single tree* menghasilkan akurasi tertinggi yaitu 0,7072 diikuti *decision tree* dengan 0,6874 dan paling kecil oleh multi *tree* dengan 0,6726.

Perolehan nilai *support genetic programming* dengan *multi tree* menghasilkan rata-rata support tertinggi yaitu 0,3854 diikuti *decision tree* dengan 0,3584 dan paling kecil *single tree* dengan 0,3494. Multi *tree* memiliki hasil *support* paling tinggi di semua jumlah data *training*. Sedangkan *single tree* memiliki *support* terendah untuk jumlah data 1626 hingga 4876. Secara umum jumlah *support* tidak sejalan dengan nilai akurasi yang dicapai.

Hasil penelitian ini jika dibandingkan dengan penelitian sebelumnya yang menggunakan decision tree yang sama untuk klasifikasi teks untuk ekstraksi kata kunci memiliki akurasi yang lebih baik dengan memanfaatkan *pre-processing map reduce* sesuai hasil di atas (Koruko, 2016). Dibandingkan dengan pemanfaatan *genetic programming* sebelumnya yang tidak menggunakan *map reduce* sebagai *pre-processing* (Tran et al., 2019), penelitian ini juga diperkirakan memiliki beban komputasi yang lebih sedikit karena memproses lebih sedikit

jenis kata. Perbandingan kinerja beban komputasi menjadi lanjutan dari penelitian ini. Temuan kami telah menghasilkan perpaduan *hadoop map reduce*, *web scraping* dan *genetic programming* dalam memproses klasifikasi teks dibandingkan dengan sebelumnya.

SIMPULAN

Penelitian ini telah mengembangkan sistem klasifikasi teks dengan *pre-processing* menggunakan *map-reduce* dan pengumpulan data menggunakan *web scraping* yang belum ada penelitian sebelumnya menggunakan algoritma *genetic programming*. Melalui *web scraping* telah dikumpulkan data dengan mengurangi duplikat sebanyak 8126. *Map-reduce* telah melakukan tokenisasi dan *stop-word removal* dengan total 28507 *term* dengan 4306 *term* unik dan 24201 term duplikasi. Evaluasi klasifikasi teks menunjukkan *single tree* menghasilkan akurasi lebih baik (0,7072) dari *decision tree* (0,6874) dan terendah adalah multi *tree* (0,6726). Untuk perolehan nilai *support genetic programming* dengan multi *tree* menghasilkan rata-rata *support* tertinggi yaitu 0,3854 diikuti *decision tree* dengan 0,3584 dan paling kecil *single tree* dengan 0,3494. Secara umum jumlah *support* tidak sejalan dengan nilai akurasi yang dicapai.

REFERENSI

Alsmadi, I., & Hoon, G. K. (2019). Term weighting scheme for short-text classification: Twitter corpuses. *Neural Computing and Applications*, *31*(8), 3819–3831. https://doi.org/10.1007/s00521-017-3298-8

Altınel, B., & Ganiz, M. C. (2018). Semantic text classification: A survey of past and recent advances. *Information Processing & Management*, *54*(6), 1129–1153. https://doi.org/10.1016/j.ipm.2018.08.001

Anjum, B. (2018). MapReduce--The Scalable Distributed Data Processing Solution. In *Topics in Parallel and Distributed Computing* (pp. 173–190). Springer, Cham.

de Sá, A. G. C., Freitas, A. A., & Pappa, G. L. (2018). Automated selection and configuration of multi-label classification algorithms with grammar-based genetic programming. In *International Conference on Parallel Problem Solving from Nature* (pp. 308–320). Springer, Cham. https://doi.org/10.1007/978-3-319-99259-4_25

Deng, X., Li, Y., Weng, J., & Zhang, J. (2019). Feature selection for text classification: A review. *Multimedia Tools and Applications*, *78*(3), 3797–3816. https://doi.org/10.1007/s11042-018-6083-5

Du, S., & Li, J. (2019). Parallel processing of improved KNN text classification algorithm based on Hadoop. *2019 7th International Conference on Information, Communication and Networks (ICICN)*, 167-170. IEEE. https://doi.org/10.1109/ICICN.2019.8834973

Halibas, A. S., Shaffi, A. S., & Mohamed, M. A. K. V. (2018). Application of text classification and clustering of Twitter data for business analytics. *Majan International Conference (MIC)*, 1-7. Oman: IEEE. https://doi.org/10.1109/MINTC.2018.8363162

Jeong, H., & Cha, K. J. (2019). An efficient mapreduce-based parallel processing framework for user-based collaborative filtering. *Symmetry*, *11*(6), 1–8. https://doi.org/10.3390/sym11060748

Koruko, S. (2016). Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, *57*, 232–247. https://doi.org/10.1016/j.eswa.2016.03.045

Kousalya, K., & Parvez, S. J. (2018). Effective processing of unstructured data using python in Hadoop map reduce. *Nternational Journal of Engineering & Technology*, *7*(2.21), 417–419. https://doi.org/10.14419/ijet.v7i2.21.12456

Koutris, P., Salihoglu, S., Suciu, D., & others. (2018). Algorithmic aspects of parallel data processing. *Foundations and Trends®in Databases*, *8*(4), 239–370. https://doi.org/10.1561/1900000055

Pintye, I., Kail, E., Kacsuk, P., & Lovas, R. (2021). Big data and machine learning framework for clouds and its usage for text classification. *Concurrency and Computation: Practice and Experience*, *33*(19), e6164. https://doi.org/10.1002/cpe.6164

Ramsingh, J., & Bhuvaneswari, V. (2018). An efficient Map Reduce-Based Hybrid NBC-TFIDF algorithm to mine the public sentiment on diabetes mellitus--A big data approach. *Journal of King Saud University-Computer and Information Sciences*, *33*(8), 1018–1029. https://doi.org/10.1016/j.jksuci.2018.06.011

Ranjitha, K. V, Prasad, B. S. V., & others. (2020). Optimization Scheme for Text Classification Using Machine Learning Naïve Bayes. In *ICDSMLA 2019* (pp. 576–586). Singapore: Springer.

Sihombing, L. O., Hannie, H., & Dermawan, B. A. (2021). Sentimen Analisis Customer Review Produk Shopee Indonesia Menggunakan Algortima Naïve Bayes Classifier. *Edumatic: Jurnal Pendidikan Informatika*, *5*(2), 233–242. https://doi.org/10.29408/edumatic.v5i2.4089

Tahmassebi, A., & Gandomi, A. H. (2018). Genetic programming based on error decomposition: A big data approach. In *Genetic programming theory and practice XV* (pp. 135–147). Springer. https://doi.org/10.1007/978-3-319-90512-9_9

Telikani, A., Gandomi, A. H., & Shahbahrami, A. (2020). A survey of evolutionary computation for association rule mining. *Information Sciences*, *524*, 318–352. https://doi.org/10.1016/j.ins.2020.02.073

Thomas, D. M., & Mathur, S. (2019, June). Data analysis by web scraping using python. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 450-454). India: IEEE. https://doi.org/10.1109/ICECA.2019.8822022

Tran, B., Xue, B., & Zhang, M. (2019). Genetic programming for multiple-feature construction on high-dimensional classification. *Pattern Recognition*, *93*, 404-417. https://doi.org/10.1016/j.patcog.2019.05.006

Viegas, F., Rocha, L., Gonçalves, M., Mourão, F., Sá, G., Salles, T., Andrade, G., & Sandin, I. (2018). A genetic programming approach for feature selection in highly dimensional skewed data. *Neurocomputing*, *273*, 554–569. https://doi.org/10.1016/j.neucom.2017.08.050