HALAMAN PENGESAHAN

PENGEMBANGAN MODEL IDENTIFIKASI PENULIS PADA DATA BIBILIOGRAFI DENGAN MENGGUNAKAN *DEEP NEURAL NETWORK* UNTUK MENINGKATKAN AKURASI DAN KEKOKOHAN

DISERTASI

Diajukan Untuk Melengkapi Salah Satu Syarat
Memperoleh Gelar Doktor Dalam Bidang Ilmu Teknik Informatika

Oleh
FIRDAUS
03043681621005

Palembang 30 Juni 2022
Promotor

Prof. Dr. Ir. Siti Nurmaini, M.T.
NIP. 196908021994012001

Koordinator Prodi

Prof. Dr. Ir. Nukman, M.T.
NIP. 195903211987031001

Mengetahui
Dekan Fakultas Teknik

Prof. Dr. Eng. Ir. Joni Arliansyah, M.T.
NIP. 196706151995121002

HALAMAN PERSETUJUAN

Karya tulis ilmiah berupa laporan disertasi ini dengan judul "Pengembangan Model Identifikasi Penulis pada Data Bibiliografi dengan Menggunakan *Deep Neural Network* untuk Meningkatkan Akurasi dan Kekokohan" telah dipertahankan dihadapan Tim Penguji Karya Tulis Ilmiah Program Studi Ilmu teknik Program Doktor Fakultas Teknik Universitas Sriwijaya pada tanggal : 6 Januari 2022

Palembang : 6 Januari 2022

Tim Penguji Karya Tulis Ilmiah berupa Laporan Disertasi

Ketua

Dr. Bhakti Yudho Suprapto., S.T., M.T.
NIP : 197502112003121002 ( )

Anggota

1. Dr. Eng. Tresna Dewi, S.T., M.Eng.
   NIP. 197711252000032001 ( )

2. Dr. Yusuf Hartono, M.Sc.
   NIP. 196411161990031002 ( )

3. Dian Palupi Rini. M.Kom.. Ph.D.
   NIP. 197802232006042002 ( )

Mengetahui
Dekan Fakultas Teknik

Prof. Dr. Eng. Ir. Joni Arliansyah, M.T.
NIP. 196706151995121002

Ketua Program Studi

Prof. Dr. Ir. Nukman, M.T.
NIP. 195903211987031001

DISERTASI

PENGEMBANGAN MODEL IDENTIFIKASI PENULIS PADA DATA BIBILIOGRAFI DENGAN MENGGUNAKAN *DEEP NEURAL NETWORK* UNTUK MENINGKATKAN AKURASI DAN KEKOKOHAN

FIRDAUS

NIM 03043681621005

PROGRAM STUDI DOKTOR ILMU TEKNIK
FAKULTAS TEKNIK
UNIVERSITAS SRIWIJAYA
TAHUN 2021

HALAMAN PENGESAHAN

DISERTASI

"PENGEMBANGAN MODEL IDENTIFIKASI PENULIS PADA DATA BIBILIOGRAFI DENGAN MENGGUNAKAN *DEEP NEURAL NETWORK* UNTUK MENINGKATKAN AKURASI DAN KEKOKOHAN"

Diusulkan oleh
Firdaus
NIM. 03043681621005

Telah disetujui
pada tanggal 20 Desember 2021

Promotor

Prof. Dr. Ir. Siti Nurmaini, M.T.
NIP. 196908021994012001

KATA PENGANTAR

Bismillahirahmanirahim

Puji syukur dipanjatkan ke hadirat Allah SWT, atas limpahan rahmat dan hidayahnya dapat menyelesaikan Proposal Disertasi dengan judul "Pengembangan Model Identifikasi Penulis pada Data Bibiliografi dengan Menggunakan *Deep Neural Network* untuk Meningkatkan Akurasi dan Kekokohan". Dalam kesempatan ini diucapkan terimakasih yang sedalam-dalamnya kepada yang terhormat:

1. Prof. Dr. Ir. Siti Nurmaini, M.T.selaku Promotor
2. Prof. H. Zainuddin Nawawi, Ph.D., selaku Ketua BKU Teknik Elektro
3. Prof. Dr. Ir. Nukman, M.T. selaku Kepala Program Studi Ilmu-Ilmu Teknik Fakultas Teknik Program Pascasarjana Universitas Sriwijaya

Saya menyadari bahwa dalam penyusunan proposal ini masih jauh dari sempurna, baik dari segi penyusunan, bahasan, ataupun penulisannya. Oleh karena itu diharapkan kritik dan saran yang sifatnya membangun, guna menjadi acuan dalam bekal pengalaman untuk lebih baik di masa yang akan datang.

Salam Hormat,

Firdaus

RINGKASAN

PENGEMBANGAN MODEL IDENTIFIKASI PENULIS PADA DATA BIBILIOGRAFI DENGAN MENGGUNAKAN *DEEP NEURAL NETWORK* UNTUK MENINGKATKAN AKURASI DAN KEKOKOHAN Karya Tulis Ilmiah Berupa Usulan Penelitian Disertasi, 14 Agustus 2020

Firdaus; dibimbing oleh Prof. Dr. Ir. Siti Nurmaini

Program Studi Ilmu Teknik Program Doktor, Fakultas Teknik/Program Pascasarjana Universitas Sriwijaya

Dengan bertambah besarnya ukuran data bibliografi pada perpustakaan digital ilmiah, menjadi tantangan untuk mengidentifikasi nama penulis dengan benar dan menempatkan publikasi tersebut kepada mereka. Kualitas data perpustakaan digital ilmiah bergantung pada proses *Author Name Disambiguation* (AND). AND mungkin terjadi karena adanya beberapa penulis dengan nama yang sama (homonim) atau variasi nama yang berbeda untuk orang yang sama (sinonim). beberapa pendekatan dilakukan untuk menyelesaikan permasalahan AND. Pendekatan untuk penyelesian masalah AND terdiri dari dua pendekatan, *author assignment* dan *author grouping*.

Penelitian ini menyoroti pendekatan *author assignment* untuk mengenali penulis sebuah publikasi. Beberapa penelitian telah dilakukan dengan menggunakan pendekatan author assignment dengan akurasi yang sangat memuaskan untuk keseluruhan data tetapi tidak terlalu memuaskan dalam *recall*. Dan juga sampai saat ini belum ada satupun penelitian yang membahas kinerja metode untuk masing-masing permasalahan AND, sehingga kinerja model tidak tergambar dengan baik. Untuk meningkatkan kinerja algoritma jaringan syaraf tiruan konvensional, sebuah DNN dengan lapisan yang banyak diusulkan pada penelitian ini, dimana struktur ini telah teruji untuk jumlah data yang sangat banyak.

Keterbaruan dari penelitian ini adalah model identifikasi penulis dengan tingkat akurasi dan kekohan yang lebih baik. Penelitian ini diharapkan menghasilkan kontribusi yang signifikan pada algoritma identifikasi penulis pada data publikasi.

Kata Kunci : Author Name Disambiguation, klasifikasi, machine learning, data bibliografi, homonim, sinonim.

DAFTAR ISI

HALAMAN PENGESAHAN ..... **ii**
KATA PENGANTAR ..... **iii**
RINGKASAN ..... **iv**
DAFTAR ISI ..... **v**
DAFTAR GAMBAR ..... **vii**
DAFTAR TABEL ..... **viii**
BAB I PENDAHULUAN **1**
1.1. Latar Belakang 1
1.2. Rumusan Masalah 3
1.3. Tujuan Penelitian 3
1.4. Sistematika Penulisan 4
BAB II TINJAUAN PUSTAKA **5**
2.1. *Author Name Disambiguation* 5
2.1.1. Permasalahan AND 6
2.1.2. Author Homonim dan Sinonim 7
2.1.3. Taksonomi AND 9
2.1.4. Dataset dan Metrik Kinerja AND 11
2.2. Pemrosesan Teks 16
2.2.1. Normalisasi Teks 17
2.2.2. Ekstraksi Fitur 30
2.2.3. Reduksi Fitur 34
2.2.4. Penskalaan dan Normalisasi Fitur 39
2.3. *Machine Learning* 39
2.3.1. *Deep Neural Network* (DNN) 39
2.3.2. SVM 44
2.4. *Performance Measurements* 49
BAB III METODOLOGI PENELITIAN **51**
3.1. Kerangka Penelitian 51

3.2.Studi Literatur 52
3.3. Persiapan Dataset 53
3.4. Pengembangan Model Identifkasi Penulis 55
3.4.1. Persiapan Data 56
3.4.2. Pra-pemrosesan Data 57
3.4.3. Klasifikasi 61
3.5. Evaluasi dan Validasi Model 63
BAB IV HASIL PENELITIAN 66
4.1. Pendahuluan 66
4.2. Pengolahan data 66
4.3. Ekstraksi Fitur 67
4.4. Reduksi Fitur 67
4.5. Evaluasi 68
4.6. Pengujian Pada Dataset Publikasi Penulis Indonesia 74
BAB V KESIMPULAN 76
6.1. Kesimpulan 76
6.2. Pengembangan Penelitian Lanjutan 76
DAFTAR PUSTAKA 77
PUBLIKASI 88

DAFTAR GAMBAR

Gambar 2.1. Komposisi data bibliografi yang terdiri dari kasus *synonym*, *homonym*, *homonym synonym*, dan *non-homonym-synonym* 8
Gambar 2.2. Taksonomi *author name disambiguation* [36] 11
Gambar 2.3. Proses *case folding* 17
Gambar 2.4. Proses *tokenizing* 18
Gambar 2.5. Proses *filtering/stopword removal* 19
Gambar 2.6. Proses *stemming* 20
Gambar 2.7. Porter stemmer algorithm 21
Gambar 2.8. Diagram alir proses *porter stemmer* 22
Gambar 2.9. Langkah *principal component analysis* (PCA) 37
Gambar 2.10. Arsitektur *neural network* 40
Gambar 2.11. Arsitektur *deep neural network* (DNN) 42
Gambar 2.12. *Support vector machine* (SVM) 46
Gambar 3.1. Kerangka penelitian pengembangan model identifkasi penulis 52
Gambar 3.2. Proses pembuatan dataset indonesia 55
Gambar 3.3. Proses persiapan data untuk identifikasi author 56
Gambar 3.4. Pra-pemrosesan data untuk identifikasi author 58
Gambar 3.5. Normalisasi atribut *title* 59
Gambar 3.6. Arsitektur DNN yang diusulkan 61
Gambar 4.1. Kinerja PCA (a) akurasi, (b) losss 68
Gambar 4.2. Kinerja DNN pada data uji 71

DAFTAR TABEL

Tabel 2.1. Dataset dan metrik kinerja yang digunakan pada beberapa penelitian[65] 12
Tabel 2.2. Step 1a-menghapus akhiran jamak 23
Tabel 2.3. Step 1b-menghapus infleksi verbal 23
Tabel 2.4. step 1b1-menghapus infleksi verbal untuk '-ed' dan '-ing' 24
Tabel 2.5. Step 1c-menghapus sufiks '*y'* dan *'i'* 24
Tabel 2.6. Step 2-mengganti satu sufiks dengan multiple sufiks 25
Tabel 2.7. Step 3-aturan penghapusan untuk multi sufiks 25
Tabel 2.8. Step 4-menghapus sufiks di akhir 26
Tabel 2.9. Step 5a-menghapus sufiks 'e' 26
Tabel 2.10. Step 5b-reduksi 27
Tabel 2.11. Perbandingan hasil *lancaster stemmer* dan *porter stemmer* 28
Tabel 2.12. Perbandingan hasil *stemmer* serta *lemmatization* 30
Tabel 2.13. Contoh *one hot encoder* (OHE) 31
Tabel 3.1. Deskripsi dataset Kim 54
Tabel 3.2. Deskripsi dataset Indonesia 54
Tabel 3.3. Metode ekstraksi fitur pada atribut-atribut bertipe teks 60
Tabel 3.4. Arsitektur DNN dan parameter tuning 62
Tabel 3.5. Perbandingan komposisi dataset asli dan dataset yang digunakan 63
Tabel 3.6. Komposisi data *training* dan data *testing* 64
Table 4.1. Perbandingan rincian jumlah dataset sebelum dan sesudah dikurangi 66
Table 4.2. Komposisi empat permasalahan AND pada dataset 67
Table 4.3. Jumlah Fitur yang dibangkitkan dari masing-masing atribut 67
Tabel 4.4. Komposisi data latih dan data uji 69
Tabel 4.5. Akurasi model (%) dari 244 struktur DNN 70
Tabel 4.6. Kinerja klasifikasi DNN 71
Tabel 4.7. Perbandingan kinerja DNN (%) dibandingkan dengan *classifier* lainnya 72

Tabel 4.8. Perbandingan kinerja DNN (%) untuk setiap permasalahan AND dibandingkan dengan *classifier* lain 73
Tabel 4.9. Jumlah fitur yang dihasilkan dari tahapan pra-pemrosesan dan reduksi fitur 74
Tabel 4.10. Perbandingan kinerja klasifikasi DNN pada data publikasi penulis Indonesia dengan dataset Kim 75

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perpustakaan digital ilmiah telah menjadi sumber penting catatan bibliografi untuk komunitas ilmiah [1], menjadi sumber penting bagi ilmuwan untuk mendapatkan literature dan menemukan topik yang menarik [2]. Perpustakaan digital ilmiah juga menyediakan analisis yang dapat digunakan untuk pengambilan keputusan yang lebih baik oleh lembaga donor dan institusi akademis menentukan penerima hibah dan promosi individu [3]. Dengan bertambah besarnya ukuran perpustakaan digital ilmiah, ini menjadi tantangan untuk mengidentifikasi nama penulis dengan benar dan menempatkan publikasi tersebut kepada mereka [4].

Kualitas data perpustakaan digital ilmiah bergantung pada proses *Author Name Disambiguation* (AND), yang mengaitkan nama penulis pada publikasi kepada orang yang tepat [5]. AND merupakan masalah penting yang perlu dipecahkan dalam analisis bibliometrik pada sebuah perpustakaan digital ilmiah [2]. AND mungkin terjadi karena adanya beberapa penulis dengan nama yang sama (homonim) atau variasi nama yang berbeda untuk orang yang sama (sinonim) [6][4].

Dalam beberapa tahun ini, beberapa pendekatan dilakukan untuk menyelesaikan permasalahan AND. Secara garis besar teknik penyelesaian AND dibagi kedalam dua kelompok [3], yaitu teknik berbasiskan machine learning yang terdiri dari supervised [7][8][9][10][11], unsupervised [12][13][14][15][16] dan semi-supervised [17][18][19][20][21][22][23][24][25] dan non machine learning yang terdiri dari teknik berbasis graph [26][27][28][29][30][31] dan teknik berbasis heuristik [32][33]. Shin et al. [27] mengajukan metode dengan menggunakan Graph Framework untuk menyelesaikan permasalahan AND, yang diselesaikan dengan proses graf termasuk pemisahan dan penggabungan vertex berdasarkan co-authorship. Namun metode ini masih belum memadai dalam kondisi minor seperti perubahan permanen pada nama atau afiliasi

seperti 'Perubahan Profil Penulis' tidak dapat ditangani secara memadai. Lin et al. [34] mengimplementasikan Hierarchical Agglomerative Clustering untuk menangani masalah AND dengan dua atribut, yaitu co-author dan atribut judul. Nama co-author dalam data dikelompokkan ke dalam beberapa kelompok, dan konsep ranking confidence diterapkan untuk mengukur confidence dari berbagai pengukuran kesamaan. Hussain and Asghar [35] menggunakan algoritma structural clustering untuk memisahkan author dengan menggunakan algoritma deteksi kelompok dan operasi graf. Sayangmya metode ini tidak mampu mendeteksi nama penulis yang sangat ambigu dalam kasus dimana salah seorang author memiliki banyak topik riset. Beberapa kelemahan dapat di eksplorasi dimasa mendatang, antara lain kutipan diri, konsep tersembunyi dan alamat email author. Pada penelitian lainnya, , Ferreira and Gonçalves [36] mengklasifikasikan pendekatan klasifikasi author dari publikasi menjadi dua kelompok, author grouping dan author assignment. Pendekatan author grouping mengelompokkan author berdasarkan kemiripan data dari atribut data publikasi, sedangkan pendekatan author assignment secara langsung menentukan sebuah publikasi dimiliki oleh seorang author dnegan membangun sebuah model yang merepresentasikan sebuah author.

Penelitian ini menyoroti pendekatan author assignment untuk mengenali author sebuah publikasi. Terdapat dua metode dalam penggalian pengetahuannya, klasifikasi dan clustering. Keunggulan dari metode klasifikasi adalah kemampuannya dalam menghadapi data publikasi yang sangat banyak untuk setiap author. Sedangkan metode clustering memerlukan informasi khusus tentang jumlah author yang sesuai atau jumlah kelas author dan metode ini memerlukan waktu yang agak panjang untuk menentukan parameternya [36]. Beberapa peneliti menggunakan pendekatan author assignment dengan klasifikasi [37][38][39]. Hasil yang didapat tidak terlalu memuaskan dalam F1-score dan akurasi [37][38], penggunaan pendekatan jaringan sayaraf tiruan telah diesplorasi untuk mengenali author dari sebuah publikasi. Namun kinerja yang dihasilkan menghasilkan recall yang kurang baik dan akurasi yang memuaskan [39].

Selain itu dari seluruh penelitian yang pernah dilakukan, belum ada satupun yang membahas hasil kinerja model berdasarkan kelompok author berdasarkan permasalahan AND, yaitu homonim dan sinonim. Berdasarkan dataset yang digunakan oleh beberapa penelitian, komposisi data author dengan kondisi homonim dan sinonim memiliki komposisi yang sedikit dibandingkan dengan data author tanpa sinonim dan homonim, dimana author data author ini

bukanlah permasalahan AND. Komposisi data demikian dapat menghasilkan akurasi yang tinggi, tetapi memiliki F1-score yang rendah.

Untuk meningkatkan kinerja algoritma jaringan syaraf tiruan konvensional, sebuah DNN dengan lapisan yang banyak diusulkan pada penelitian ini. DNN memiliki kemampuan yang sangat baik dalam mempelajari fitur dalam pekerjaan yang banyak dan dapat memecahkan permasalahan authorship [10]. DNN dapat membangun model yang dapat memecahkan masalah ambiguasi nama author secara bertahap ketika sebuah data publikasi diintegrasikan dalam dataset. Penelitian ini juga menyelidiki kinerja model AND terhadapat empat kategori permasalahan AND [40], homonim, sinonim, homonim-sinonim, dan non-homonim-sinonim. Sebagai pembanding, *classifier* SVM, Random Forest, Naïve Bayes, dan decision Tree digunakan sebagai pembanding.

1.2. Rumusan Masalah

Pertanyaan riset :

1. Bagaimana memodelkan identifikasi author pada data bibliografi
2. Bagaimana membuat system model yang dirancang sesuai dengan kinerja yang diharapkan
3. Melakukan analisis dari model yg dipilih dalam hal efisiensi dan efektifitas model machine learning yang dirancang

1.3. Tujuan Penelitian

Tujuan utama dari penelitian ini adalah:

a) Mengembangkan model identifkasi penulis pada data biiliografi
b) Merancang algoritma identifkasi penulis dengan menggunakan teknik machine learning
c) Mengevaluasi hasil identifikasi dalam hal tingkat akurasi dan *recall*
d) Memvalidasi model yang dihasilkan untuk mencapai kekokohan (*robustnes*)

Pertanyaan riset:

Bagaimana membuat model identifikasi author yang dapat berkerja secara akurat dan efisien pada data bibliografi

1.4. Sistematika Penulisan

Proposal penelitian ini terdiri dari empat bab. Bab pertama menjelaskan tentang latar belakang masalah perlunya penelitian ini dilakukan mengacu pada penelitian-penelitian yang pernah dilakukan sebelumnya. Bab kedua menjelaskan tentang penelitian-penelitian sebelumnya yang terkait dengan AND, terutama dengan pendekatan *author assigment*, pemrosesan teks yang terdiri dari normalisasi teks dan ekstraksi fitur dari teks, reduksi fitur dan teknik klasifikasi dengan menggunakan pendekatan *machine learning*. Bab tiga menjelaskan tentang metodologi penelitian yang digunakan untuk menyelesaikan permasalahan-permasalahan yang sudah dijabarkan sebelumnya dalam beberapa tahapan yang terstruktur untuk mencapai tujuan yang diharapkan. Sedangkan bab lima berisi kesimpulan sementara dari penelitian pendahuluan yang telah dilakukan untuk mengetahui apakah penelitian ini layak untuk dilakukan.

BAB II
TINJAUAN PUSTAKA

2.1. *Author Name Disambiguation*

Disambiguasi adalah proses mengidentifikasi arti kata dalam suatu kalimat upaya untuk menentukan makna yang tepat dari sebuah kata yang ambigu. Disambiguasi nama penulis adalah hal yang sangat penting untuk membedakan antara orang-orang yang memiliki nama yang sama di mana pengenal unik tidak ada. Dalam banyak aplikasi, disambiguasi nama orang adalah hal yang penting untuk keberhasilan aplikasi. Dalam pencarian web, diperkirakan bahwa 5-10% dari seluruh *traffic* pencarian web adalah queri nama orang yang lebih baik disajikan ketika hasil tidak ambigu.

Pada Umumnya, ambiguitas nama orang datang dalam tiga varietas; 1) masalah aliasing: ketika seseorang menggunakan beberapa variasi nama seperti "Ronald W. Williams" dan "R.W. Williams ", 2) masalah nama umum: ketika ada lebih dari satu orang dengan nama yang sama, yang terutama bermasalah dengan frekuensi tinggi nama seperti nama-nama Cina, dan 3) kesalahan tipografi yang dapat dihasilkan dari input manusia atau sistem ekstraksi otomatis yang terjadi.

Author name disambiguation (AND) dapat membantu dalam banyak kasus; salah satunya pengguna mencoba melakukan pencarian dalam semua artikel yang ditulis oleh penulis tertentu. Disambiguasi juga memungkinkan analisis bibliometrik yang lebih baik dengan memungkinkan penghitungan dan pengelompokan publikasi dan kutipanyang lebih akurat. Selain itu, dalam proses membangun perpustakaan digital sangat penting untuk mengidentifikasi seseorang yang menulis artikel tertentu dan menemukan semua publikasinya. Biasanya informasi yang diberikan oleh publikasi tidak cukup untuk secara tepat mengindikasikan seorang penulis karya tersebut.

Pada kasus yang lain, ketika mengekstrak informasi tentang peneliti, aktivitas dan publikasi, disambiguasi nama adalah salah satu hal yang amat penting. Biasanya, ada seperangkat publikasi dengan teks dan serangkaian penelitian yang mewakili nama penulis tertentu (dan

terkadang dengan atribut tambahan yang membentuk profil peneliti) ketika ingin memberikan setiap publikasi ke salah satu peneliti masalahnya rumit karena biasanya ada berbagai bentuk nama peneliti, misalnya sebuah nama dapat ditulis dengan nama belakang dan terakhir dari seorang peneliti (dengan atau tanpa nama tengah), atau dengan inisial untuk nama pertama, dengan atau tanpa inisial Untuk nama tengah. Sering terjadi bahwa nama pertama dan terakhir penulis berasal dari kamus nama pertama, dalam hal ini nama keluarga secara keliru dianggap sebagai nama pertama (dan kemudian diganti dengan inisial). Dan yang terakhir, untuk nama populer itu sangat sering terjadi bahwa dua orang yang berbeda memiliki pasangan yang sama (nama depan, nama keluarga).

2.1.1. Permasalahan AND

Terdapat empat faktor utama yang mendasari permasalahan pada AND [41]. Permasalahan pertama adalah keadaan apabila seorang penulis (*author*) memublikasikan dokumen penelitiannya dan nama yang digunakannya berbeda-beda. Permasalahan tersebut meliputi; 1) nama serta ejaan nama yang ditulis beragam, 2) pada saat penulisan nama dan ejaan terdapat ketidaktepatan (*human error*), 3) kehidupan yang mengalami perubahan dan menyebabkan perubahan pada nama penulis (*author*), misalnya ikatan dalam pernikahan, berpindah agama, dan atau sebab lainnya, 4) penulis yang tidak mencantumkan nama aslinya atau menggunakan nama alias (samaran).

Permasalahan kedua adalah persamaan nama yang dimiliki oleh orang yang berbeda (*homonym*). Kenyataannya, pada beberapa kasus banyak individu mempunyai nama sama dikarenakan nama tersebut umum untuk digunakan. Kebalikannya, seorang penulis (*author*) tidak memiliki satu nama saja akan tetapi dapat memiliki nama yang bervariasi (*synonym)*, penyebab dari hal ini dikarenakan sebagaimana yang telah dijabarkan di paragraf sebelumnya [42] [43].

Permasalahan ketiga adalah ketidaklengkapan metadata (data yang memiliki informasi dalam hal menunjukkan, menggambarkan, mendapatkan, serta menyajikan informasi tentang data lain) tentu saja sangat diperlukan agar dapat melakukan prosedur disambiguasi nama, misalnya tidak terdapatnya nama depan penulis (*author*) pada sebagian database publikasi serta bibliografi, lokasi geografi, serta keterangan lebih lanjut mengenai pengenalan penulis misalnya titel ataupun kedudukan yang dimiliki oleh penulis [44].

Permasalahan keempat adalah melonjaknya persentase pada manuskrip publikasi ilmiah yang bukan hanya sebatas di penulis yang lebih dari satu, namun meliputi lembaga (instansi) yang lebih dari satu serta akademik yang berpartisipasi untuk suatu publikasi ilmiah. Empat hal yang telah disebutkan merupakan hal yang mendasari agar dapat dilakukannya pemecahan untuk persoalan disambiguasi nama penulis (*author*).

2.1.2. Author Homonim dan Sinonim

Pada aspek-aspek persoalan AND, aspek kedua yang berkaitan dengan *homonym* serta *synonym* ialah aspek yang sangat penting serta menjadi pemicu utama alasan timbulnya suatu persoalan ambiguitas nama [45]. Pada sebuah database bibliografi biasanya terdiri dari empat kombinasi permasalahan (gambar 2.1), *synonym* (persamaan 1) dan *homonym* (persamaan 2), kombinasi diantara keduanya, *homonym-synonym* (persamaan 3), dan yang bukan keduanya, *non-homonym-synonym* (persamaan 4).

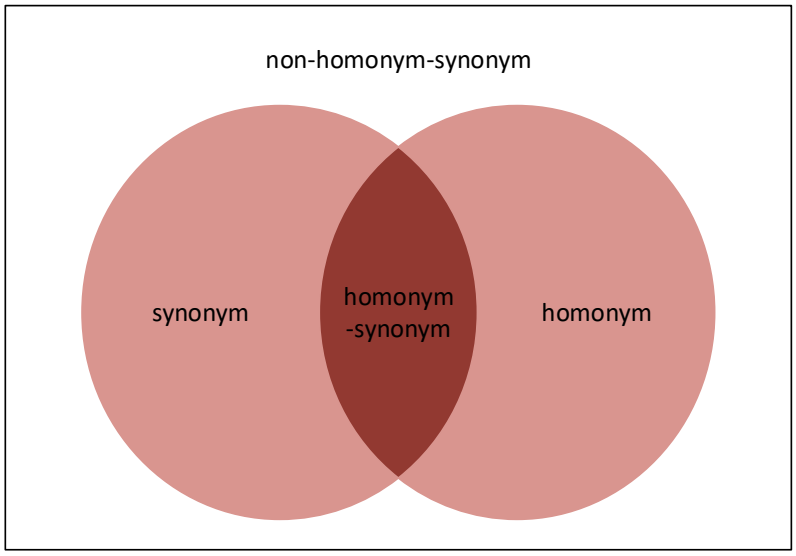
$$homonym: X \rightarrow Y \quad 1 \mapsto m, \qquad m \geq 2 \tag{1}$$

$$synonym = Y \rightarrow X \quad 1 \mapsto n, \qquad n \geq 2 \tag{2}$$

$$where, \qquad X = presented\ name \quad Y = author$$

$$homonymsynonym = homonym \cap synonym \tag{3}$$

$$nonhomonymsynonym = (homonym \cup synoniym)^{C} \tag{4}$$



Gambar 2.1. Komposisi data bibliografi yang terdiri dari kasus *synonym*, *homonym*, *homonym synonym*, dan *non-homonym-synonym*.

Dalam melaksanakan pengenalan (identifikasi) serta klasifikasi penulis tersebut merupakan penulis yang sesuai atau tidak sesuai, aspek homonym serta synonym lah yang paling berpengaruh. Suatu perpustakaan digital, keadaan *homonym* serta *synonym* tersebut dapat mengakibatkan hasil keluaran yang tidak tepat, tidak sesuai sasaran serta tidak tepat untuk hasil keluaran yang sedang ditargetkan [35]. *Polysems* atau keadaan *homonym* mengakibatkan [10] [36] [27] [46], luaran yang memberikan hasil informasi yang tidak sesuai ke penulis yang mempunyai persamaan nama. Sebaliknya keadaan *synonym*, paling sukar untuk melakukan pengumpulan informasi penerbitan yang memiliki keterkaitan di bagian nilai maksimum *recall* (perbandingan dari nilai benar positif dengan keseluruhan data yang benar positif) yang tidak memberikan informasi data yang utuh pada beragam nama yang dipakai oleh seseorang penulis untuk penerbitannya [45].

Persoalan *homonym* memiliki contohnya sebagai berikut, diketahui terdapat satu orang penulis yang memiliki nama “D. Johnson”, apabila nama tersebut dicari maka merujuk pada beraneka ragam nama, seperti “David B. Johnson” dengan asal venue Lecture University, “David S. Johnson” dengan asal venue DS&T Research Lab, atau “David E. Johnson” dengan asal venue Brain University. Persoalan *synonym* contohnya ialah, satu orang penulis memiliki nama “David

S. Johnson", apabila nama tersebut dicari maka merujuk pada beraneka ragam nama, apabila nama penulis dibuat lebih pendek seperti "David Johnson", "D. Johnson", contoh lain "D. S. Johnson" studi kasus lainnya dapat juga disaat kondisi penulisan serta pengejaan nama penulis yang salah diantaranya "Davad Johnson". Faktor-faktor tersebut mengakibatkan timbulnya nama yang ambigu [47].

Dari contoh kasus tersebut dapat simpulkan maka ambiguitas nama ialah persoalan penting yang harus ditemukan jalan penyelesaiannya dikarenakan memiliki keterkaitan pada ketelitian dan keakurasian data yang menjadi sumber untuk suatu informasi yang sangat diperlukan [45].

2.1.3. Taksonomi AND

Penelitian sebelumnya yang telah banyak dilakukan, penyelesaian untuk persoalan AND ada terdapat banyak teknik serta pendekatan yang dapat dipakai [48] [49] [50]. Kesimpulan yang didapatkan dari banyaknya metode pada berbagai hasil penelitian ialah terdapat tiga pendekatan yang bisa dipakai agar dapat menjadi jalan penyelesaian persoalan AND, yaitu *supervised*, *unsupervised*, serta *semi-supervised* [51].

Pendekatan *supervised* merupakan salah satu metode disambiguasi [47] data latih diwajibkan telah mempunyai label atau telah diberikan tanda, yang nantinya dapat digunakan, lalu memperoleh model klasifikasi yang unggul menyesuaikan dengan seberapa besar percobaan yang akan dikerjakan pada data latih tersebut. Percobaan yang akan banyak dilakukan akan menghasilkan satu model klasifikasi yang unggul. Lalu, hasil terbaik dipergunakan agar dapat melakukan pengenalan (identifikasi) dan mengambil kesimpulan akankah satu orang penulis terbaru (data terbaru) serta penulis pada data latih ialah penulis yang merupakan orang yang sama atau berbeda (tidak sama).

Data latih yang telah memiliki label, pada *supervised learning* digunakan untuk melakukan prediksi ataupun klasifikasi [52]. Contoh penerapan prosedur pemecahan *supervised learning* adalah pengenalan angka tulisan tangan. Dalam kasus ini, dikumpulkan ribuan gambar angka tulisan tangan serta dengan label yang berisikan angka sebenarnya agar dapat menjadi perwakilan di setiap gambar. Selanjutnya kaitan diantara gambar serta angka (label) akan dipelajari oleh algoritam serta mengimplementasikan hasil dari analisa sebelumnya agar dapat dilakukan

pengklasifikasian gambar terbaru (tidak memiliki label atau label baru) yang belum pernah diamati maupun dipelajari oleh sistem pada saat prosesnya. Sedangkan tidak adanya penggunakan data latih dalam proses prediksi ataupun klasifikasi algoritma *unsupervised learning* melainkan adanya penggunaan informasi serta data penunjang untuk mengkalkulasikan seberapa serupa antara dua atau lebih objek yang mungkin serupa bahkan hampir sama untuk satu golongan atau kelompok algoritma [53] [51].

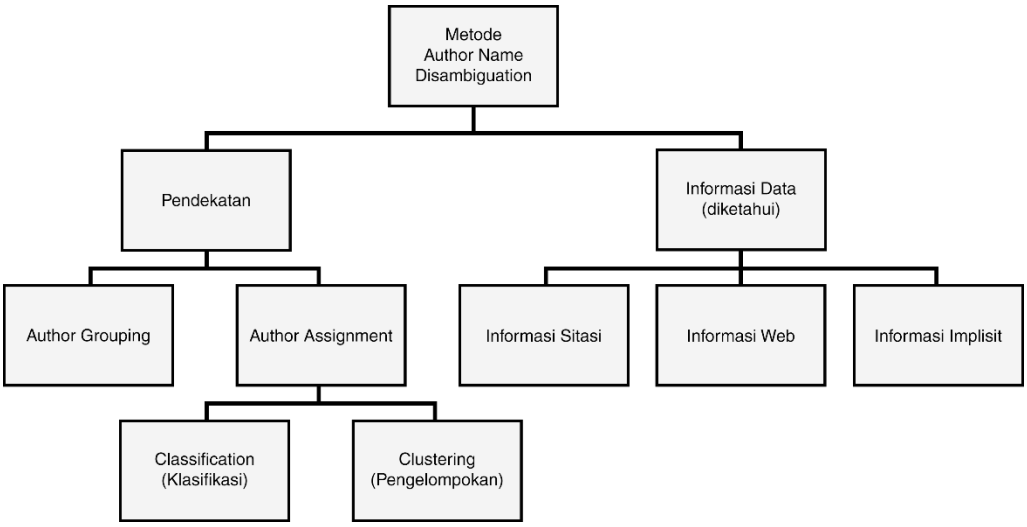
Hal yang membedakan diantara kedua teknik tersebut adalah pada label [20]. Data yang telah mempunyai suatu kelas tertentu disebut dengan data berlabel, kebalikannya data yang belum memiliki kelas disebut dengan data yang tidak berlabel. Pada pengimplementasiannya, masing-masing teknik ini memerlukan seluruh informasi atau disebut *feature* sebagai dasar dalam self-learning serta percobaan data untuk pemakaian dari metode pendekatannya [52].

Pendekatan *unsupervised* metode disambiguasinya adalah dengan [54][50] adanya penggunaan data imformasi tambahan/informasi pendukung agar dapat dikalkulasi seberapa kesamaannya antar publikasi lalu setelahnya algoritma clustering akan digunakan untuk proses pemecahan persoalan disambiguasi nama penulis (*author*). Pendekatan *semi-supervised* metode disambiguasinya adalah dengan [15] [55] nama lainnya adalah pendekatan *hybrid* [56] ialah kombinasi metode dari *supervised* serta *unsupervised*. Penelitian yang dilakukan Ferreira [37] serta Torvik [57] dalam hal ini dilakukan pendekatan *Hybrid*, penerapan metode heuristik (pemeriksaan serta mengumpulkan sumber) agar dapat dihasilkan suatu set referensi yang memiliki label secara otomatis dan setelahnya memiliki kegunaan sebagai data latih untuk metode *supervised*.

Ferreira [36] menekankan jika terdapat dua buah taksonomi hierarkis yang menjadi pendekatan untuk persoalan author name disambiguation dan kerap kali oleh peneliti sebelumnya digunakan pada pengelompokan secara otomatis, diantaranya Pengelompokan Author (*Author Grouping)* dan *Author Assignment*. Pendekatan Pengelompokan Author (*Author Grouping*) mengelompokkan seberapa mirip dan sama data dari publikasi penulis serta pendekatan *Author Assignment* suatu publikasi akan langsung ditentukan kepemilikannya.

Penelitian Ferreira, Goncalves, dan Laender [36] menjelaskan perbedaan bahwasanya metode yang dapat digunakan untuk penyelesaian permasalahan AND pada saat keadaan *Synonym*

ialah *Author Grouping* serta penyelesaian pada keadaaan *Homonym* permasalahan AND dapat menggunakan metode *Author Assignment*. Penelitian setelahnya, Hussain dan Ashger [58] telah melakukan diskusi serta membandingkan metode AND yang para peneliti pakai semenjak dari tahun 2010 dan telah ditentukan apa keunggulan serta kekurangan untuk setiap metode tersebut. Penggunaan struktur taksonomi yang berbeda sebagaimana *Graph-Based, Machine Learning*, dan sebagainya mereka gunakan. Penelitian yang dilakukan, Ferreira [36] memberikan penjelasan suatu taksonomi AND yang menjadi dasar suatu solusi permasalahan AND. Apabila dibuat gambar pada suatu struktur hirarki, gambar 2.2 memperlihatkan strutur hirarki taksonomi AND.



Gambar 2.2. Taksonomi *author name disambiguation* [36]

2.1.4. Dataset dan Metrik Kinerja AND

Beragam dataset yang digunakan pada penelitian sebelumnya, baik dataset hasil ekstraksi Digital Library maupun dataset sintetik. Adapun Digital library yang digunakan sebagai sumber diantaranya adalah Brazilian Digital Library of Computing (BDBComp), Digital Bibliography and Library Project (DBLP), Arnetminer, Microsoft Academic Search (MAS), CiteSeer, dan beberapa penelitian menggunakan dataset yang dikembangkan oleh mereka sendiri, seperti dataset China, Korea, Jerman Vietnam dan lain-lain.

Dataset BDBComp digunakan dalam penelitian yang dilakukan antara lain oleh Carvalho[12], Ferreira[17] [36], Veloso[59], dan Santana[60]. Sedangkan dataset DBLP antara lain digunakan dalam penelitian Momeni[4], Tran[9], Carvalho[12], Peng[22], Ferreira[17] [61], Liu[13], Tang[15], Smalheiser [41], Andruszkiewicz [62], Imran[19], Veloso[59], Lee[26], Shin[27], Peng[22], Santana[60], Levin[20], Zhu[25], Yang[63], dan Han[7]. Dataset Arnetminer digunakan dalam penelitian yang dilakukan oleh Liu[13], Tang,[15], dan Shin[27]. Dataset MAS digunakan oleh Liu[64], Chin[32], Zhao[24], dan Andruszkiewicz [62]. Dataset CiteSeer digunakan oleh Tang[15] dan Andruszkiewicz [62].

Para peneliti telah mengevaluasi AND pada perpustakaan digital dengan berbagai macam data berlabel dan berbagai macam metrik kinerja. Tabel 2.1. memperlihatkan beberapa penelitian berikut dataset yang digunakan dan metrik kinerja yang digunakan.

Tabel 2.1. Dataset dan metrik kinerja yang digunakan pada beberapa penelitian[65]

Penelitian	**Library**	**Ukuran Data**	**Jumlah data berlabel**	**Metrik Kinerja**
Reijnhoudt dkk [66]	SCOPUS (*general*)	45 M	Sebanyak 57.775 paper yang berasal dari 1400 profesor di Belanda tepatnya National Academic Research and Collaboration Information System (NARCIS)	Presisi dan Sensitivity Baseline: N/A Mean
Kawashima dan Tomizawa [67]	SCOPUS (*general*)	N/A	Sebanyak 573,338 paper yang terkait dengan 75,405 peneliti dari Jepang mendapatkan bantuan hibah untuk melakukan Penelitian Database Ilmiah (KAKEN)	Rata-rata Presisi dan Sensitivity Baseline: N/A
Torvik dan Smalheiser [57]	PubMed (*medicine*)	15.3 M	(1) 62 nama penulis diacak dari berbagai pasangan paper (2) Sebanyak 20.085 profil penulis di Community of Science (COS) (3) Sebanyak 2.313 profil penulis yang paling banyak di sitasi di ISI	*Lumping & Splitting Errors* Baseline: SCOPUS, WOS

			(4) 83.992 *PI groups in NIH funding database* (5) 323,274 *self-citation pairs* (6) Sebanyak 148 Sampel acak dari profil penulis di COS	
Liu dkk [45]	PubMed (*medicine*)	22 M	(1) 300 *stratified random pairs* (2) Sebanyak 40 profl penulis yang paling banyak disitasi (3) Sebanyak 47 profil peneliti NIH (4) 4.7 M *self-citation pairs* (5) 23 M *grant-citation pairs*	(Rata-rata) *Pairwise* Presisi/Spesifisity/ F1-Score; *Splitting Error Baseline*: Torvik and Smalheiser (2009)
Lerchenmueller dan Sorenson [68]	PubMed (*medicine*)	15.3 M	Sebanyak 355.921 paper yang terkait dengan 36.987 id penyelidik utama di NIH ExPORTER	Rata-rata Presisi dan Spesifisity Baseline: N/A
Levin dkk [20]	Web of Science (*general*)	14 M	Sebanyak 15.750 paper dari 237 penulis telah melakukan konfirmasi daftar publikasi nya lewat email	Rata-rata *Pairwise* Presisi/Spesifisity/ F1-Score Rata-rata B-Cubed Presisi/Spesifisity/ F1-Score Baseline: *Unsupervised Agglomerative Clustering*
Schulz dkk [14]	Web of Science (*general*)	47 M	(1) Tautan publikasi Google Scholar-WOS untuk 3.000 nama keluarga (2) 138 *random publication pairs* (3) Sebanyak 4.7 M Kelompok nama yang berisi setidaknya dua nama dengan inisial nama depan kedua (4) Sebanyak 26.887 kluster dari 3.000 pasangan nama acak (5) Sebanyak 110.011 kluster email diekstraksi dari arXiv	Rata-rata Presisi dan Spesifisity Baseline:
Martin dkk [69]	APS (*physics*)	460 K	Sebanyak 79 pilihan acak dari pasangan penulis yang cocok	*Merging & Splitting Error*

			dan 111 yang tidak cocok dengan nama yang mirip	*Baseline: raw name*
Sinatra dkk [70]	APS (*physics*)	450 K	Sebanyak 200 pilihan acak dari pasangan penulis yang cocok dan 200 yang tidak cocok dengan nama yang mirip	*Merging & Splitting Error* Baseline:
Wang dkk (2011) [30]	AMiner (*computing*) 450	1.6 M	Sebanyak 6730 paper dari 1382 penulis dengan nama yang sama	Rata-rata Pairwise Presisi/Spesifisity/ F1-Score Baseline: *DISTINCT, SA-Cluster CONSTRAINT*, HAC
Louppe dkk [21]	INSPIRE (*high- energy physics*) 1	1 M	Sebanyak 360.066 paper dari 36.340 penulis yang diidentifikasi oleh penulis sebenarnya, kurator, serta penerbit	Rata-rata Pairwise Presisi/Spesifisity/ F1-Score Rata-rata B-Cubed Presisi/Spesifisity/ F1-Score *Baseline: first-initial method*

Beragam varian dari publikasi (seperti karya tulis ilmiah, jurnal, esai, dan sebagainya) yang tersedia begitu banyak di Digital Library (DL) dan di dalamnya berisi banyak informasi penting (seperti informasi mengenai penulis). Pada penelitian sebelumnya, kurang lebih para peneliti menggunakan 18 buah Digital Library, diantaranya; AMiner/Arnet Miner (computing) [58] [71], PubMed (biomedicine) [72] [73] [45] [74], Web of Science [20], CiteSeerX [75] [56], DBLP (computer science) [44] [47] [25] [42], BDBComp [58], SCOPUS [73], ACM/Association for Computing Machinery, IEEE Xplore/Institute of Electrical and Electronics Engineers, Microsoft Academic Search [10], Google Scholars, Astrophysics Data System (ADS), ACM Portal, arVix, MEDLINE [73], Libra (Academic Search), RePEc, dan INSPIRE (high-energy physics) [46].

Sumber informasi dari beragam komunitas bidang ilmu komputer yang merupakan isi dari dataset DBLP, serta terdapat berbagai macam publikasi paper yang tergabung dalam satu wilayah penelitian yang sama [44]. Dataset DBLP ialah dataset yang terstruktur serta mempunyai bentuk data yang terbaik [76] penelitian yang berpokok untuk menangani persoalan AND akan sangat cocok jika menggunakan dataset DBLP sebagai dataset penelitiannya. Walaupun data termasuk ke dalam data yang sederhana dan rapi, proses pembersihan data (*cleaning process*) harus tetap dilakukan agar dataset lebih disempurnakan lagi [25]. D. Shin et al. melakukan penelitian [27] pembersihan data (*cleaning process*) pun dilakukan pada dataset AND berbasis DBLP yang

digunakan untuk penelitian. Proses pembersihan tersebut ialah proses perbaikan dataset. Hal yang dilakukan dalam perbaikan ialah penghilangan data-data yang eror, data yang *double* (lebih dari satu), serta data yang samar-samar (tidak jelas), kemudian akan dilakukan pengisian kembali untuk bagian yang kosong digantikan dengan data terbaru yang telah disempurnakan.

Telah begitu banyak dilakukan penelitian yang memiliki fokus pada hal ekstraksi data dengan tujuan agar dapat terciptanya suatu dataset baru terutama untuk penelitian AND. Maka banyak juga dataset lain yang digunakan oleh peneliti. Dataset KANG merupakan salah satunya, peneliti Kang et al. dari Korea yang menciptakannya [77] peneliti Santana et al. pernah menggunakan dataset tersebut [33] untuk penelitian mengenai disambiguasi pula. Terdapat 41.673 data nama penulis serta dokumen publikasi yang telah diekstraksi dari DBLP pada dataset KANG. Dataset KANG menandai sekitar 6921 label data nama penulis yang berbeda.

Dataset TANG yang dibuat oleh peneliti Tang et al. yang berasal dari Tsinghua University di Cina yang merupakan contoh dari dataset lainnya [15]. Tujuan penciptaan Dataset TANG ialah melatih algoritma disambiguasi pada Digital Library AMiner. Sebanyak 7.528 data nama penulis yang memiliki sumber dari AMiner yang terdapat pada dataset TANG dan sekitar 110 nama dapat memenuhi untuk kasus disambiguasi. Terdapat 1.546 label pada dataset TANG yang menjadi perwakilan untuk tiap nama penulis yang berbeda.

Peneliti dari Cina spesifiknya dari Xi'an Jiaotong University menciptakan dataset QIAN yang menjadi contoh dari dataset lainnya peneliti Qian et al. [78] serta peneliti lainnya dari Jepang spesifiknya Waseda University. Gabungan antara dataset The GILES dan AMiner yang telah dilakukan pengkoreksian merupakan isi dari dataset QIAN. Terdapat 6783 nama penulis pada dataset QIAN dan sekitar 574 kasus nama yang ambigu. Seperti dataset DBLP, dataset-dataset ini ialah dataset labeled data maksudnya data tersebut telah mempunyai label data, label tersebut adalah proses manual yang dilakukan oleh peneliti untuk dataset tersebut [65].

Sekitar 1.200.000 paper yang berasal dari beragam *venue* (lokasi publikasi) pada fokus ilmu komputer yang merupakan cakupan dari dataset DBLP serta membuatnya menjadi website bibliografi ilmu komputer terbesar [25] [79]. Pada dataset DBLP, pengidentifikasian penulis sesuai dengan nama penulis [76]. Santana [33] di tahun 2015, sebanyak 8.414 sitasi dataset DBLP digunakan yang telah selaras dengan 477 penulis yang berbeda, dalam hal ini di kisaran rata-rata

17,6 sitasi untuk seorang penulis. Dataset AND pada banyak penelitian telah sering menjadikan DBLP sebagai sumber data nya. Dataset AND berbasis DBLP telah banyak dilakukan perubahan oleh para peneliti menyesuaikan dengan metode yang digunakan. Hal ini menyebabkan banyaknya dataset AND berbasis DBLP yang lebih beragam. Seperti penelitian yang dilakukan Han et al. [44] hasil penelitiannya pada modifikasi dataset DBLP diberikan nama Han-DBLP, selanjutnya penelitian yang dilakukan Qian et al. [78] dataset DBLP hasil modifikasinya diberikan nama Qian-DBLP, serta penelitian yang dilakukan Kang et al. [77] dataset DBLP hasil modifikasinya diberikan nama KISTI-AD-E-01. Menyesuaikan dengan tujuan para peneliti dalam menciptakan dataset berbasis DBLP sesuai dengan modifikasinya masing-masing.

Pada umumnya dataset AND berbasis Digital Library memiliki kesamaan, ialah sama-sama mempunyai informasi pada sebuah publikasi bersama dengan feature-feature yang ada juga pada dataset AND yang lain, misalnya Venue, Title, Year, dan lain lain. Hal yang menjadi pembeda antar dataset adalah format data pada dataset tersebut, seberapa aktualnya data dengan publikasi yang baru, data yang memiliki informasi jelas serta lengkap, dan besar ukuran atau jumlah data yang dimiliki dataset. Karena hal tersebut, dataset dengan berbagai kelemahan atau kelebihan yang dimiliki oleh masing-masing dataset, tetap terus-terusan dilakukan penelitian dengan tujuan agar dapat menghasilkan dataset yang terbaik untuk dijadikan sumber data penelitian selanjutnya.

2.2. Pemrosesan Teks

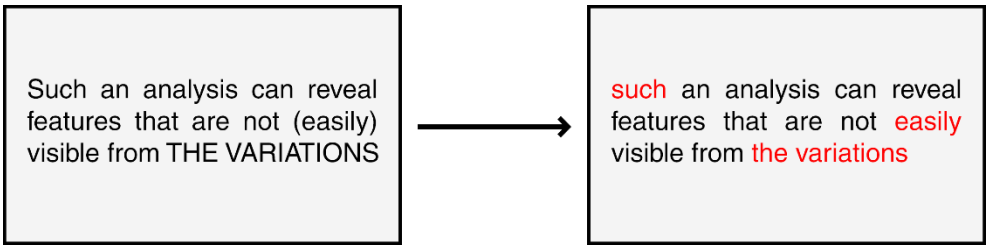
Text Processing yang mencakup *Natural Language Processing* (NLP) adalah langkah pertama yang akan dilakukan sebelum melanjutkan ke tahapan utama yaitu klasifikasi. *Text Processing* ini memiliki fungsi untuk melakukan persiapan data yang akan digunakan akan lebih tersusun serta efisien untuk diolah ke dalam tahap selanjutnya. *Natural Language Processing* adalah suatu teknologi kecerdasan buatan (AI) yang berfungsi agar dapat mengubah data utama di satu dokumen dalam format text menjadi satu data kuantitatif dalam bentuk angka (*numerical form*) dengan cara cepat. Maka, metode serta pendekatan yang nanti dipakai akan lebih efisien pada data yang dikerjakan. Data yang telah berbentuk angka akan lebih mudah bagi yang akan menggunakan nantinya serta memudahkan *classifier* disebabkan data lebih sederhana agar dapat diproses oleh program dibandingkan menggunakan data yang berbentuk teks.

2.2.1. Normalisasi Teks

Normalisasi Teks adalah tahapan yang didalamnya terdapat sebuah proses yang disebut dengan *Data Cleaning. Data Cleaning* adalah suatu proses pembersihan data-data yang terdapat didalam dataset yang akan digunakan. Pembersihan ini bertujuan agar data yang akan digunakan dapat diolah lebih mudah pada tahapan berikutnya. Lebih rinci, proses *Data Cleaning* adalah tahapan untuk melakukan pengidentifikasian data-data yang memiliki karakter atau simbol yang tidak jelas dan tidak berkaitan dengan esensi data tersebut. Setelah dilakukan identifikasi, maka simbol atau karakter tersebut akan dihapus atau dihilangkan. Dan tidak hanya itu, proses pada tahapan *Data Cleaning* juga melakukan perbaikan terhadap data-data yang tidak lengkap/kosong (null), data-data yang salah, serta data yang tidak akurat atau tidak relevan untuk diproses pada tahap selanjutnya. Dengan menggunakan bahasa pemrogrman pyhton dnegan platform NLTK, beberapa alat bantu dapat digunakan untuk proses normalisasi teks.

2.2.1.1. Case Folding

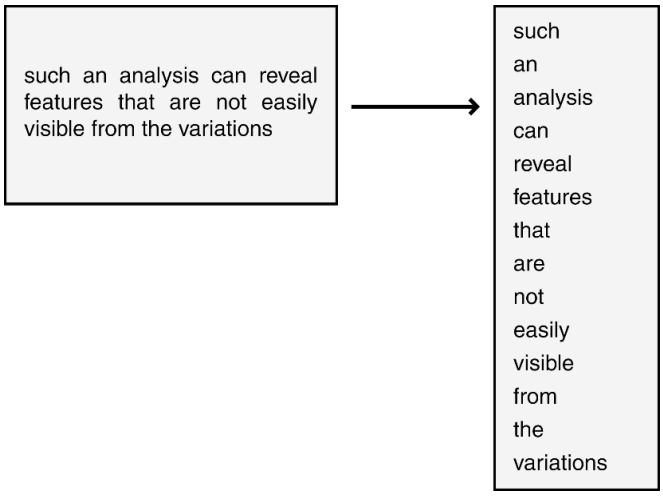
Case Folding adalah sebuah proses yang dilakukan untuk merubah bentuk huruf (bukan bentuk teks). Pada proses ini, setiap huruf untuk semua kalimat yang terdapat didalam dataset akan dirubah menjadi huruf kecil saja tanpa ada huruf kapital. Selain daripada itu, pada proses ini juga semua karakter dan simbol yang bukan bagian dari kalimat (data penting) akan dihapuskan kecuali *delimiter*/pembatas. Secara sederhana, perubahan yang dilakukan pada proses ini dapat diperhatikan di Gambar 2.3.



Gambar 2.3. Proses *case folding*

2.2.1.2. Tokenizing

Tokenizing adalah proses lanjut dari *case* folding. Pada proses ini, setelah semua huruf menjadi sama rata dalam bentuk huruf kecil maka semua kata yang terbentuk oleh huruf-huruf itu akan dipisahkan dari kalimat-kalimatnya. Sebuah kalimat yang tersusun atas kata-kata akan menghilang menjadi hanya satu kata saja. Secara sederhana, kata-kata tersebut dapat dianggap dipisahkan per-kotak/blok. Contoh perubahan yang terjadi dari proses *Tokenizing* ini dapat diperhatikan di Gambar 2.4.

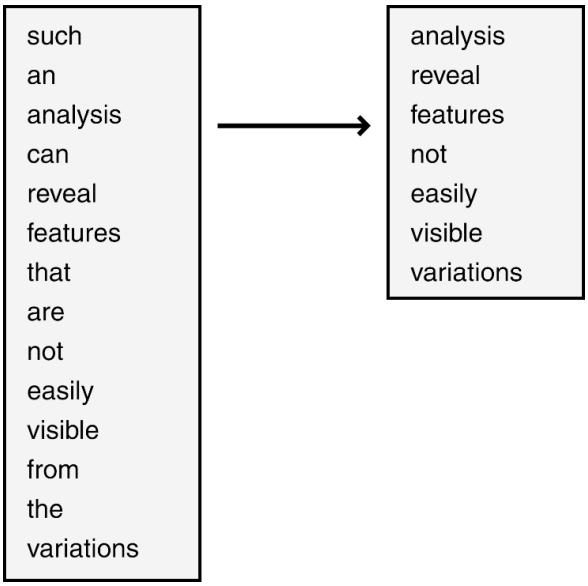


Gambar 2.4. Proses *tokenizing*

2.2.1.3. Filtering/Stopword Removal

Filtering atau *Stopword Removal* adalah proses lanjutan dari *Tokenizing*. Setelah data yang berbentuk teks atau kalimat telah dipisahkan per-kata pada proses *Tokenizing*, selanjutnya adalah proses pemilihan kata (*filtering*). Pada proses ini, kata-kata yang telah dipisahkan akan dipilih dan diambil hanya kata-kata yang memiliki arti saja (penting) serta sekaligus mengeliminasi kata-kata yang tidak memiliki arti (*stopword removal*). Kata penting/memiliki arti disini ialah kata yang mempunyai makna atau keterangan serta bukan merupakan konjungsi dan imbuhan. Pemilihan kata ini adalah proses yang sangat penting untuk dilakukan. Hal ini dikarenakan inti data yang diperlukan adalah kata-kata atau teks yang beresensi dan dapat menjadi bahan perhitungan pada

pemrosesan (kata yang memiliki pengaruh besar). Dalam kasus ini, kata yang memiliki pengaruh besar tentu bukan kata sambung, melainkan sebuah kata yang memiliki arti dan dapat didefinisikan. Pemilihan kata yang hanya mengambil kata yang penting akan sangat meningkatkan efisiensi pemrosesan. Alasannya adalah mesin akan mudah mempelajari kata-kata dengan bentuk yang sama sehingga proses yang terjadi akan lebih cepat dan klasifikasi ataupun identifikasi dapat dilakukan dengan mudah oleh model yang dibangun. Selain daripada itu, cepatnya pemrosesan juga sangat dipengaruhi oleh banyak data yang diolah. Dengan menghilangkan kata sambung dan imbuhan, maka secara otomatis beban komputasi akan berkurang dan hanya mengolah kata yang penting saja. Sederhananya, proses *filtering* dan *stopword* dapat diperhatikan di Gambar 2.5.

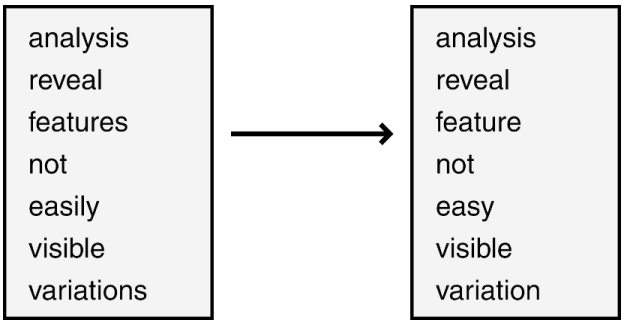


Gambar 2.5. Proses *filtering/stopword removal*

2.2.1.4. Stemming

Stemming merupakan sebuah teknik yang prosesnya mirip dengan *stopword*. Namun, perbedaannya adalah *stemming* menghilangkan imbuhan, sedangkan *stopword* menghilangkan kata. Pada proses *stemming*, setiap kata yang telah difilter akan dikembalikan kedalam bentuk kata dasar/baku tanpa imbuhan. Seperti yang telah dijelaskan pada sub-bab *Filtering/Stopword Removal*, penghilangan imbuhan ini bertujuan untuk membuat proses yang dilakukan menjadi lebih efisien. Dengan hilangnya imbuhan pada setiap kata yang akan digunakan, maka proses

komputasi yang dilakukan mesin akan menjadi lebih ringan dan lebih cepat. Contoh perubahan data yang dilakukan *stemming* dapat diperhatikan di Gambar 2.6.

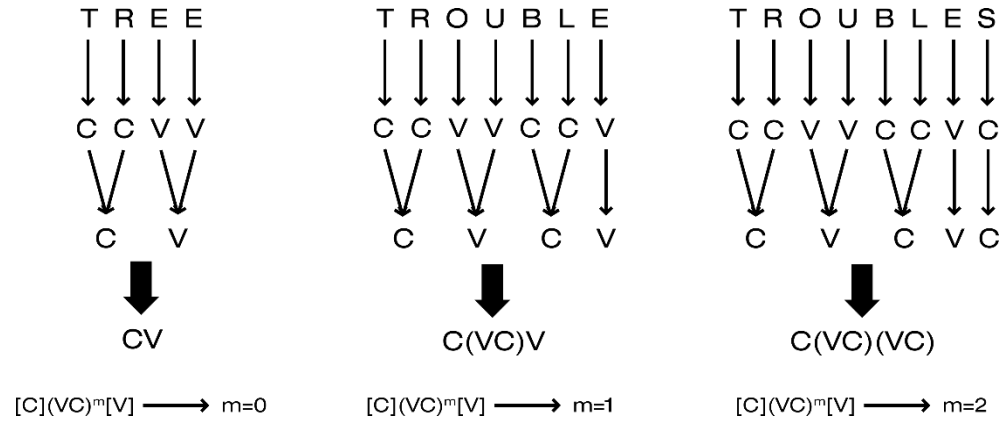


Gambar 2.6. Proses *stemming*

Stemming dalam prosesnya sangat dipengaruhi oleh bahasa yang digunakan didalam dataset (bahasa penulisan data didalam dataset). Hal ini dikarenakan proses *stemming* adalah proses yang menyangkut tata cara penulisan (*Grammar*). Tanpa pengetahuan tentang bahasa yang digunakan untuk kata yang akan dibakukan, maka proses *stemming* tidak bisa dilakukan karena tidak diketahuinya yang mana merupakan kata asli dan yang mana merupakan imbuhan. Beruntungnya, pada penelitian ini dataset yang digunakan adalah dataset yang menggunakan bahasa inggris. Beberapa tools yang dapat digunakan untuk melakukan *stemming* (dengan untuk bahasa yang dataset berbeda-beda) adalah *Porter Stemmer*, *Snowball Stemmer*, *Lancaster Stemmer*, *Lovins Stemmer*, *Husk/Paice Stemmer*, *Krovetz Stemmer*, serta algoritma yang digunakan pada proses *Stemming* lainnya.

Porter Stemmer merupakan sebuah tools *stemming* yang dapat digunakan untuk melakukan *stemming* terhadap dataset yang berbahasa inggris yang cocok untuk digunakan dalam penelitian ini. Penamaan *porter stemmer* untuk tools ini sesuai dengan nama pembuatnya, yaitu Martin F. Porter. Martin Porter mengatakan bahwa terciptanya *tools* ini dengan tujuan agar tidak terdapatnya kata yang multi-makna yang disebabkan oleh imbuhan kata tersebut. Imbuhan pada suatu kata dapat menjadikan kata tersebut menjadi berbeda makna. Dalam pemrosesan teks dengan ML, imbuhan kata dapat memperumit komputasi karena bertambahnya jumlah data (karakter) yang diolah. Selain itu, imbuhan kata juga dapat menyebabkan kurang efektifnya proses *learning* yang dilakukan mesin, karena banyak terdapat kata yang mirip (akibat dari imbuhan). Secara sederhana,

porter stemmer akan berkerja terhadap data dengan cara mendeteksi kata dasar didalam kalimat pada data tersebut, lalu menghilangkan/menghapus imbuhannya. Hal ini dapat terjadi oleh karena algoritma *porter stemmer* yang dibangun telah banyak belajar tentang grammar imbuhan kata dalam bahasa inggris. Pola algoritma *Porter Stemmer* dapat diperhatikan di Gambar 2.7.



Gambar 2.7. Porter stemmer algorithm

Dari gambaran algoritma *porter stemmer* yang diperlihatkan di Gambar 2.7, keterangan pertama yang didapatkan adalah: variable C adalah huruf yang mewakili kata *Consonants* (huruf mati) dan variable V huruf yang mewakili kata *Vowels* (huruf vokal). Didalam bahasa inggris, huruf konsonan adalah semua huruf didalam alfabet diluar dari huruf vokal serta huruf "Y" (apabila pada suatu kata didahului oleh huruf konsonan) yang artinya huruf "Y" bisa menjadi huruf konsonan atau huruf vokal sesuai kondisi. Misal huruf "Y" didalam kata "PLAY", dalam kata ini huruf "Y" adalah huruf konsonan sama seperti huruf "P" dan huruf "L". Tetapi, jika kondisi huruf "Y" seperti pada kata "REALLY", maka huruf "Y" menjadi huruf vokal sama seperti huruf "E" dan huruf "A". Sedangkan huruf Vokal adalah semua huruf didalam alfabet kecuali huruf Konsonan, yaitu huruf "A", "E", "I", "O", dan "U". Terdapat beberapa kondisi dan batasan khusus pada algoritma *Porter Stemmer*:

a. *Measure,* huruf *m* yang mewakilkan, suatu stem berdasarkan dengan susunan huruf konsonan serta vokal.
   m = 0, misalnya; FR, EE, FREE, Y, MY

m = 1, misalnya; TROUBLE, EAST, FREED, IVY

m = 2, misalnya; TROUBLES, PRIVATE, OATEN

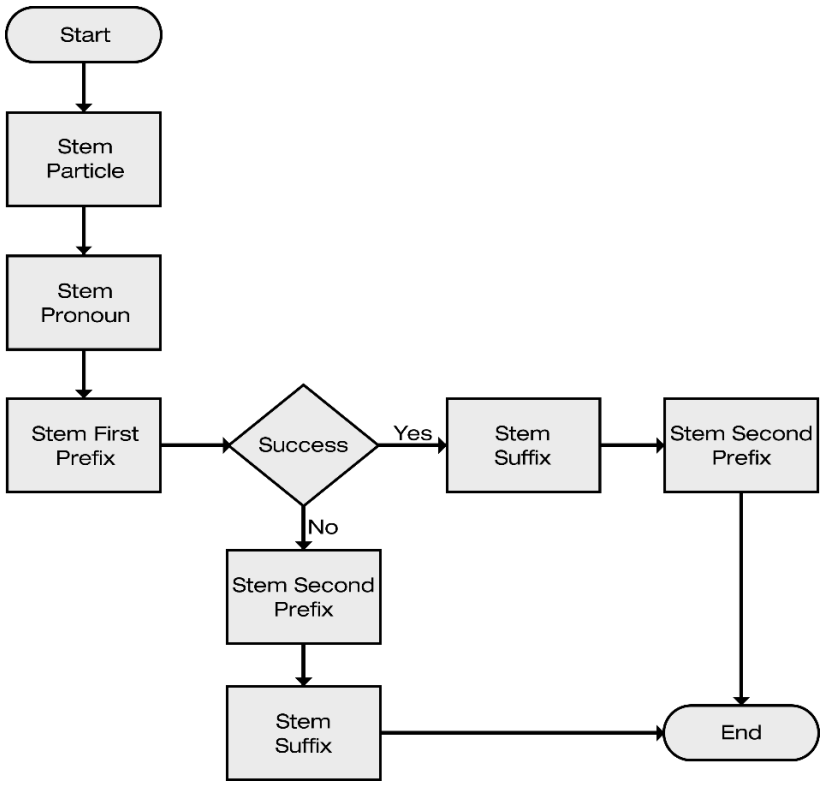
b. *<X> = penggunaan huruf X di akhir stem

c. *v* = stem yang memuat huruf vokal

d. *d = di akhir stem akan memuat konsonan lebih dari satu

e. *o = urutan C-V-C di bagian akhir stem yang berurutan serta di bagian konsonan akhir tidak termasuk huruf "W", "X", maupun "Y" (misalnya -WET, -HOP).

Langkah-langkah prosedur yang *Porter Stemmer* lakukan dapat dilihat pada Gambar 2.8.



Gambar 2.8. Diagram alir proses *porter stemmer*

Dapat diperhatikan pada Gambar 2.8. adalah sebuah *flow diagram* proses *stemming* yang dilakukan oleh *Porter Stemmer*. Proses tersebut dimulai dengan *stem particle* atau pengambilan

kata yang akan di-*stemming*, dan langsung berlanjut pada *stem pronoun* atau melihat kata dasar/baku dari kata tersebut. Kemudian algoritma *porter stemmer* akan langsung melihat/mendeteksi kata imbuhan yang menempel pada kata dasar tersebut dan langsung menghilangkannya yang dimulai dengan imbuhan pada awal kata (*prefix*). Jika terdapat imbuhan pada awal kata dan telah dihilangkan maka algoritma ini akan terus berlanjut mendeteksi dan menghilangkan imbuhan pada akhir kata. Namun, jika tidak ditemukan imbuhan pada awal kata tersebut, maka algoritma ini akan lanjut melihat dan memperbaiki kata yang baru/yang lain.

Selain dari batasan, *Porter Stemmer* juga memiliki beberapa *rules* atau aturan didalam pengambilan keputusannya. *Rules* ini biasa disebut dengan istilah *5-Steps Rules*. Dalam penerapannya, *rules* ini harus dilakukan secara berurutan dan teratur karena setiap *rule* memiliki fungsi yang berkesinambungan antar *rule* yang lain. Berikut *5-Steps Rules* nya antara lain:

a. Step 1a, sufiks (imbuhan pada akhir kata) dihilangkan dari kata yang menunjukkan lebih dari satu (jamak) menyesuaikan aturan (*rules*) seperti di tabel 2.2.

Tabel 2.2. Step 1a-menghapus akhiran jamak

Kondisi	Sufiks	Penggantian	Contoh
NULL	sses	Ss	*guesses -> guess*
NULL	ies	I	*chilies -> chili* *lies -> lie*
NULL	ss	Ss	*guess -> guess*
NULL	s	NULL	*cows -> cow*

b. Step 1b, infleksi (kata baru yang terbentuk dengan penambahan imbuhan) dihilangkan untuk kata-kata verbal berdasarkan landasan aturan (*rules*) seperti di tabel 2.3.

Tabel 2.3. Step 1b-menghapus infleksi verbal

Kondisi	Sufiks	Penggantian	Contoh
(m > 0)	eed	Ee	*treed -> tree* *seed -> see*
(*v*)	ed	NULL	*searched -> search* *seasoned -> season*
(*v*)	ing	NULL	*seating -> seat* *selling -> sell*

c. Step 1b1, kata-kata verbal yang mempunyai akhiran *-ed* serta *-ing* dihilangkan infleksi nya berdasarkan aturan (*rules)* sesuai di tabel 2.4.

Tabel 2.4. step 1b1-menghapus infleksi verbal untuk '-ed' dan '-ing'

Kondisi	Sufiks	Penggantian	Contoh
NULL	at	Ate	*populat(ed) -> populate*
NULL	bl	Ble	*nobl(ing) -> noble*
NULL	iz	Ize	*friz(ed) -> frize*
(*d and not (*<L> or *<S> or *<Z>))	null	single letter	*fell(ing) -> fell* *dress(ed) -> dress* *egg(ing) -> egg* *discuss(ing) -> discus* *cross(ed) -> cross*
(m=1 and *o)	null	E	*creat(ing) -> create* *creas(ing) -> crease*

d. Step 1c, kata-kata yang pada bagian akhir (sufiks) memiliki huruf *y* diganti menjadi *i* menyesuaikan dengan aturan (*rules*) berdasarkan tabel 2.5.

Tabel 2.5. Step 1c-menghapus sufiks '*y'* dan *'i'*

Kondisi	Sufiks	Penggantian	Contoh
(*v*)	Y	I	*university -> universiti* *apply -> appli*

e. Step 2, suatu kata dihilangkan serta diganti sufiksnya menggunakan multi sufiks menyesuaikan aturan (rules) berdasarkan tabel 2.6.

Tabel 2.6. Step 2-mengganti satu sufiks dengan multiple sufiks

Kondisi	Sufiks	Penggantian	Contoh
(m > 0)	ational	ate	*appreciational -> appreciate*
(m > 0)	tional	tion	*communicational -> communication* *calculational -> calculation*
(m > 0)	enci	ence	*containenci -> containence*
(m > 0)	anci	ance	*dependanci -> dependance*
(m > 0)	izer	ize	*sanitizer -> sanitize*
(m > 0)	abli	able	*differabli -> differable*
(m > 0)	alli	al	*expectalli –> expectal*
(m > 0)	ently	ent	*forgetently -> forgetent*
(m > 0)	eli	e	*generateli -> generate*
(m > 0)	ousli	ous	*hearousli -> herous*
(m > 0)	ization	ize	*digitalization -> digitalize*
(m > 0)	ation	ate	*hesitation -> hesitate*
(m > 0)	ator	ate	*illustrator -> illustrate*
(m > 0)	alism	al	*informalism -> informal*
(m > 0)	iveness	ive	*insensitiveness -> insensitive*
(m > 0)	fulness	ful	*knowfulness -> knowful*
(m > 0)	ousness	ous	*learnousness -> learnous*
(m > 0)	aliti	al	*maintainaliti -> maintainal*
(m > 0)	iviti	ive	*negotiativiti -> negotiative*
(m > 0)	biliti	ble	*receivabiliti -> receivable*

f. Step 3, sufiks suatu kata diganti serta dihilangkan menggunakan multi-sufiks pada kata yang telah ditentukan menyesuaikan aturan (rules) di tabel 2.7.

Tabel 2.7. Step 3-aturan penghapusan untuk multi sufiks

Kondisi	Sufiks	Penggantian	Contoh
(m > 0)	Icate	ic	*elasticate -> elastic*
(m > 0)	ative	NULL	*relative -> rel*
(m > 0)	alize	al	*realize -> real*
(m > 0)	iciti	ic	*elasticiti -> elastic*
(m > 0)	ical	ic	*elastical -> elastic*
(m > 0)	ful	NULL	*remainful -> remain*
(m > 0)	ness	NULL	*sadness -> sad*

g. Step 4, sufiks di bagian akhir dihapus menyesuaikan aturan sebenarnya dari *Porter Stemmer* dapat dilihat penjelasannya di tabel 2.8.

Tabel 2.8. Step 4-menghapus sufiks di akhir

Kondisi	Sufiks	Penggantian	Contoh
(m > 1)	al	NULL	*sufferal -> suffer*
(m > 1)	ance	NULL	*throwance -> throw*
(m > 1)	ence	NULL	*difference -> differ*
(m > 1)	er	NULL	*warner -> warn*
(m > 1)	ic	NULL	*correctic -> correct*
(m > 1)	able	NULL	*securable -> secur*
(m > 1)	ible	NULL	*frequentible -> frequent*
(m > 1)	ant	NULL	*contestant -> contest*
(m > 1)	ement	NULL	*slightement -> slight*
(m > 1)	ment	NULL	*placement -> place*
(m > 1)	ent	NULL	*confident -> confid*
(m > 1)	ion	NULL	*dissension -> dissens*
(m > 1)	ou	NULL	*laterou -> later*
(m > 1)	ism	NULL	*racism -> rac*
(m > 1)	ate	NULL	*hardate -> hard*
(m > 1)	iti	NULL	*realiti -> real*
(m > 1)	ous	NULL	*betterous -> better*
(m > 1)	ive	NULL	*straightive -> straight*
(m > 1)	ize	NULL	*brightize -> bright*

h. Step 5a, kata yang memiliki akhiran huruf '*e*' dihilangkan menyesuaikan dengan aturan berdasarkan penjelasan di tabel 2.9.

Tabel 2.9. Step 5a-menghapus sufiks 'e'

Kondisi	Sufiks	Penggantian	Contoh
(m > 1)	e	NULL	*wilde -> wild* *thine -> thin*
(m = 1 and not *o)	e	NULL	*theme -> them*

i. Step 5b, kata yang mengandung akhiran huruf yang sama/*double* direduksi menyesuaikan pada penjelasan di tabel 2.10.

Tabel 2.10. Step 5b-reduksi

Kondisi	Sufiks	Penggantian	Contoh
(m = 1 and *d and *<L>)	NULL	satu huruf	*across -> acros* *add -> ad*

Tidak hanya *Porter Stemmer*, terdapat juga *stemmer* yang lain dengan model yang sama, yaitu *Lancaster Stemmer*. Persamaan utama antara *Porter Stemmer* dan *Lancaster Stemmer* adalah pada fungsinya yang diperuntukkan untuk data/teks yang berbahasa inggris. Peneliti Chris Paice serta Gareth Husk yang membuat dan mengembangkan *Lancaster Stemmer*, mulai memperkenalkannya pada tahun 1990 (setelah adanya *porter stemmer*). *Lancaster Stemmer* diperkenalkan pertama kali adalah dengan nama yang sesuai dengan nama pembuatnya, yaitu *Paice/Husk Stemmer*. Meskipun dibuat oleh orang yang berbeda, namun inspirasi lahirnya *Lancaster Stemmer* tidak lepas dari pendahulunya, yaitu *porter stemmer*. Secara teknis, *Lancaster Stemmer* dikembangkan berdasarkan pembelajaran terhadap *porter stemmer* yang kemudian dimodifikasi dengan pendekatan yang lebih baik didalam pengolahan data dan prosesnya. Sumber yang sama, tidak membuat cara kerja dan performa kedua *stemmer* ini sama karena sudah modifikasi. Dengan demikian, kedua *stemmer* ini memiliki kelebihan dan kekurangan masing-masing.

Dengan cara kerja, performa, serta kelebihan dan kekurangan masing-masing, *porter stemmer* dan *Lancaster Stemmer* akan menghasilkan data hasil *stemming* yang berbeda. Contoh hasil yang berbeda pada proses *Lancaster Stemmer* serta *Porter Stemmer* dapat diperhatikan di Table 2.11.

Tabel 2.11. Perbandingan hasil *lancaster stemmer* dan *porter stemmer*

Kata	Porter Stemmer	Lancaster Stemmer
relation	*relation*	*relation*
relationship	*relationship*	*relation*
relations	*relation*	*relation*
relationships	*relationship*	*relation*
duckbil	*duckbil*	*duckbl*
destabilize	*destabil*	*Dest*
planting	*plant*	*plant*
railroad	*railroad*	*railroad*
moonlight	*moonlight*	*moonlight*
bell	*bel*	*bel*

Dengan *rules* yang dimilikinya, *porter stemmer* terlihat melakukan *stemming* dengan hasil yang berbeda dengan *Lancaster Stemmer* (untuk beberapa kata). *Stemming* yang dilakukan oleh *porter stemmer* dengan *rules*-nya tampak hanya menghilangkan sedikit imbuhan (suffix) saja dan hanya sewajarnya dan (sebagian besar) tidak mengembalikan kata-kata kedalam bentuk bakunya/dasar. Dengan begitu, hasil stemming yang dilakukan oleh *porter stemmer* memiliki perbedaan yang sedikit sekali antara kata yang belum dan sudah di-*stemming*. Sedangkan proses *stemming* yang dilakukan *Lancaster Stemmer* tampak lebih agresif dan menghasilkan bentuk kata yang kembali ke kata dasar/baku (untuk beberapa kata). Dengan *rules* sebanyak 120 *rule*, *Lancaster Stemmer* melakukan proses *stemming* dengan penghapusan imbuhan (suffix) yang lebih banyak sehingga untuk beberapa kata benar-benar kembali ke bentuk baku. Tidak seperti *porter stemmer*, *Lancaster Stemmer* menggunakan *rules*-nya tidak berdasarkan step, melainkan berdasarkan kebutuhan yang dilihat dari kondisi kata yang akan di-*stemming*. Setiap rules-nya memiliki fungsi dan cara kerja masing-masing untuk melakukan proses *stemming*. *Lancaster Stemmer* akan mengabaikan kata-kata yang hanya berisikan dua atau tiga huruf saja.

Kelebihan dari proses *stemming* yang dilakukan oleh *Lancaster Stemmer* adalah *flexible*-nya *rule* yang bisa digunakan. Dengan *rules* yang lebih teratur dan sederhana, *Lancaster Stemmer* mampu melakukan proses *stemming* dengan tepat dan dengan hasil yang lebih baik daripada *porter stemmer* dan ini sangat berpengaruh pada efisiensi proses *stemming*. Namun, kekurangan dari algoritma yang agresif (yang melakukan penghapusan imbuhan yang lebih banyak) adalah terjadinya proses *over-stemming* yang akan mengakibatkan hilangnya makna kata yang di-*stemming*. Sedangkan pada *Porter Stemmer*, *over-stemming* tidak akan terjadi karena *rules* yang

dimiliki oleh *porter stemmer* tidak membuat algoritma *stemming* menjadi agresif. Namun, kekurangannya adalah size data hasil *stemming* yang lebih besar karena pengaruh hasil *stemming*-nya yang tidak terlalu jauh berbeda dengan kata sebelum di-*stemming* (hanya menghilangkan sedikit imbuhan). Kedua stemmer ini memiliki kelebihan dan kelemahannya masing-masing yang bergantung pada *rules* yang dimilikinya.

2.2.1.5. Lemmatization

Lemmatization adalah sebuah teknik atau proses yang tidak jauh berbeda dengan stemming. Tujuan dan maksud yang ingin dicapai dari kedua proses ini adalah sama, yaitu penyederhanaan sebuah kata yang akan diproses lebih lanjut (dijadikan *fitur/input*). Yang berbeda dari kedua proses ini adalah terletak pada cara yang digunakan untuk mencapai tujuan tersebut. Pada proses *stemming*, langkah yang dilakukan untuk mengembalikan sebuah kata menjadi bentuk bakunya/dasar adalah dengan cara menghilangkan/menghapuskan prefix (imbuhan awal) dan suffix (imbuhan akhir) pada kata tersebut dengan memanfaatkan fungsi *stemmer*. Pada proses ini, efisiensi untuk mendapatkan bentuk baku sebuah kata dengan sempurna sangat kurang karena yang dilakukan algoritma *stemmer* adalah menghapus ujung atau akhir sebuah kata tanpa mengetahui makna katanya. Sedangkan pada proses *Lemmatization* langkah yang dilakukan untuk mengembalikan sebuah kata kedalam bentuk bakunya tidak demikian. Algoritma yang dibangun pada proses *Lemmatization* tidak memotong awal ataupun ujung kata yang diinginkan, akan tetapi benar-benar menerjemahkan kata tersebut kedalam bentuk bakunya. Misal terdapat sebuah kata comes, coming, dan came yang pada proses *Lemmatization* akan dirubah kedalam bentuk bakunya, yaitu come. Hal ini berlaku untuk semua kata dan untuk setiap kata yang sama. *Lemmatization* mampu memberikan hasil pemrosesan teks yang lebih baik daripada *Stemming*. Hasil pemrosesan teks yang baik akan sangat mempengaruhi *output* dari proses yang akan dilakukan berikutnya (klasifikasi, identifikasi, segmentasi, dan sebagainya). *Lemmatization* mampu memberikan hasil nilai presisi yang lebih unggul daripada *Stemming*, walaupun nilai hasil perbandingannya tidak terlalu signifikan [80]. Contoh perbedaan hasil *Stemming* dan *Lemmatization* dapat dilihat pada Table 2.12.

Tabel 2.12. Perbandingan hasil *stemmer* serta *lemmatization*

Form	***Stemming***	***Lemmatization***
satisfies	*satisfi*	*satisfy*
satisfying	*satisfi*	*satisfy*
sees	*see*	*see*
seeing	*see*	*see*
saw	*saw*	*see*
beautifully	*beautiful*	*beautiful*
lovely	*lovely*	*beautiful*
pretty	*pretty*	*beautiful*
broughts	*brought*	*bring*

2.2.2. Ekstraksi Fitur

Data berbentuk teks tidak dapat digunakan sebagai masukan dalam *machine learning*. Masukan *machine learning* harus berbentuk numerik. Oleh karena itu dibutuhkan sebuah cara untuk mengubah teks menjadi fitur yang berbentuk numerik. Terdapat beberapa pendekatan dalam ekstraksi fitur teks, diantaranya adalah model probabilitas, frekuensi dokumen, *information gain*, *mutual information*, rasio probabilitas *chi-square statistic* dan lainnya [81].

2.2.2.1. *One Hot Encoder* (OHE)

One Hot Encoder (OHE) adalah sebuah teknik dengan proses pengolahan data yang akan merubah bentuk data yang semula teks menjadi angka yang merepresentasikan data teks tersebut. Ini adalah proses yang bertujuan untuk membuat data input/fitur yang akan dimasukkan kedalam model ML menjadi relevan dan dapat dengan mudah dimengerti oleh model/mesin. Transformasi bentuk data ini merupakan proses yang penting untuk dilakukan karena menyangkut kelangsungan proses berikutnya (klasifikasi, identifikasi, segmentasi, dan sebagainya). Dari beberapa penelitian, proses transformasi data dengan menggunakan teknik OHE memperlihatkan hasil yang sangat baik. Selain itu, dengan pemrosesan data OHE durasi *training* untuk satu model DNN pada proses klasifikasi teks akan mengalami peningkatan secara eksponensial bersamaan dengan kosa kata (*vocabulary*) dari data [82].

Salah satu preferensi yang sangat baik untuk suatu model klasfikasi teks adalah dengan menggunakan DNN, disebabkan oleh tingkat kemampuan yang dimiliki oleh DNN yang baik dalam mempelajari fitur dan pola yang sukar, serta tidak memerlukan *domain knowledge* yang lebih rinci serta tanpa perlu melibatkan manusia. Banyak penelitian sebelumnya, DNN dapat memberikan hasil nilai akurasi yang sangat tinggi pada beragam klasifikasi teks serta pengaplikasiannya pada NLP [83] [84]

Data input yang berbentuk teks akan sangat sulit untuk diproses dan diolah oleh model/mesin. Maka dari itu, disinilah peran penting OHE untuk dapat menerjemahkan data yang berbentuk teks menjadi data yang berbentuk angka. Perubahan bentuk data yang dilakukan oleh OHE tidak akan merubah esensi dan inti dari data tersebut karena data *numerical* yang dibuat adalah data yang mencerminkan data teks yang sebenarnya. Dengan data yang *numerical*, model/mesin akan lebih mudah untuk mengolah dan melakukan klasifikasi, identifikasi, dan atau lain sebagainya. Hasil dari proses yang terjadi adalah data yang berbentuk angka yang merepresentasikan teks yang diproses dengan nilai 0 (nol) atau 1 (satu) yang biasa disebut dengan *dummy variable*.

Tabel 2.13. Contoh *one hot encoder* (OHE)

Categorical Variable	***Dummy A***	***Dummy B***	***Dummy C***
A	1	0	0
A	1	0	0
B	0	1	0
A	1	0	0
B	0	1	0
C	0	0	1
A	1	0	0

Pada penerapannya, proses OHE akan membuat sebuah *dictionary* sendiri untuk dapat menentukan nilai (1 atau 0) didalam sebuah kolom atau baris. Proses ini akan menjadikan data input menjadi lebih besar dimensinya dengan nilai yang hanya 0 atau 1. Dimensi data menjadi lebih lebar karena setiap baris data akan dijadikan kolom tanpa menghilangkan baris data asal. Jika ditemukan sebuah kata atau angka yang sesuai dengan *Dictionary*, maka nilainya akan 1 dan selain daripada itu adalah 0. Meskipun dimensi data menjadi lebih besar, tetapi bentuk data yang seperti

ini akan lebih mudah diolah dan dimengerti oleh model/mesin dibandingkan data yang berbentuk teks. Secara sederhana, contoh hasil dari proses yang dilakukan OHE dapat dilihat pada tabel 2.13.

2.2.2.2. *Term Frequency* (TF)

Term Frequency (TF) adalah sebuah teknik yang dapat digunakan untuk melakukan perhitungan terhadap jumlah sebuah kata yang tertulis didalam sebuah dokumen [85]. Algoritma yang dibangun didalam TF ini akan bekerja menghitung seberapa banyak jumlah sebuah kata muncul didalam sebuah dokumen. Dari proses ini, keluaran yang akan dihasilkan adalah sebuah nilai yang disebut dengan bobot. Tentu saja, perhitungan oleh TF ini sangat berkaitan dengan jumlah Dokumen yang ada sehingga hasil yang akan didapatkan adalah jumlah sebuah kata terhadap / didalam beberapa dokumen (nilai bobot didalam beberapa dokumen). Proses yang terjadi dalam algoritma TF, dapat dilihat untuk rumusnya pada persamaan (2.1.).

$$W(d,t) = TF(d,t) \tag{2.1.}$$

W merupakan *Weight* atau bobot yang akan menjadi nilai keluaran dari proses perhitungan yang terjadi yang mana meliput jumlah *term* (t) terhadap *document* (d). $TF(d,t)$ adalah *Term Frequency* atau perhitungan terhadap *term* t yang terdapat didalam *Documents* d. TF sangat penting pada pemrosesan teks untuk melakukan klasifikasi. Secara teknis, penerapan pemrosesan teks dengan TF akan dikombinasikan dengan DF (*Document Frequency*) yang akan menjadi sebuah teknik pemrosesan teks yang biasa disebut dengan TF-IDF. Seperti yang telah dijelaskan sebelumnya, kombinasi ini adalah mutlak karena keterkaitan antara *Term* dan *Document*.

2.2.2.3. *Document Frequency* (DF)

Seperti yang telah disinggung pada pembahasan sebelumnya, *Document Frequency* (DF) adalah sebuah teknik yang berkaitan dengan *Term Frequency* (TF) yang fungsinya juga adalah sama, namun dengan objek yang berbeda. Secara sederhana, DF melakukan tugas dengan algoritma yang mirip dengan TF yang tujuannya adalah untuk menghitung jumlah *Document* yang memuat suatu kata (kata yang telah dihitung oleh TF). Kaitan inilah yang menjadikan TF dan DF merupakan suatu kombinasi teknik yang mutlak. Sebagai penjelasan sederhana, TF dan DF adalah

dua buah bagian dari suatu teknik besar didalam proses transformasi data yang disebut dengan TF-IDF dan akan dibahas pada sub-bub berikutnya.

Dari proses yang dilakukan oleh algoritma DF, sebuah kata yang sudah memiliki nilai TF (didalam proses TF) juga akan mendapatkan sebuah nilai / *value* DF untuk setiap kata yang terdapat didalam seluruh dokumen. Seperti yang sudah dijelaskan sebelumnya, misal sebuah kata "*base*" memiliki nilai TF 100 dan nilai DF 10, maka secara sederhana nilai TF terhadap nilai DF adalah 100:10 (seratus per sepuluh) yang artinya adalah terdapat seratus kata didalam sepuluh *Document*. Nilai TF dan DF suatu kata akan diperhitungkan / digunakan jika nilai tersebut merupakan nilai yang kaitannya saling mendukung dan akan berpengaruh besar pada proses berikutnya [85]. Contoh nilai yang tidak dapat diperhitungkan adalah jika terdapat sebuah kata "*base*" yang memiliki nilai TF 100 dan memiliki nilai DF 1, setelahnya juga terdapat sebuah kata lain yang memiliki nilai TF 30 dan nilai DF juga 1, maka kedua kata ini tidak dapat diperhitungkan karena hanya terdapat didalam sebuah dokumen yang sama.

Pada DF, disaat proses pengelompokan dokumen, *Term* yang apabila tingkat munculnya kecil tidak mempunyai pengaruh yang terlalu besar. Jika DF nilainya di bawah *threshold* yang telah ditetapkan, maka suatu *term* akan disingkirkan. Fungsi dari penyingkiran *term* ini akan berkurangnya dimensi fitur yang besar pada saat melakukan pemrosesan pengolahan teks. DF adalah suatu metode seleksi fitur (*feature selection*) yang tidak rumit serta durasi pengkomputasian yang digunakan rendah.

2.2.2.4. TF-IDF

Seperti yang telah disinggung pada sub-bab sebelumnya, setelah penjelasan tentang *Term Frequency* (TF) dan *Document Frequency* (DF), yang berikutnya adalah *Term Frequency – Inverse Document Frequency* (TF-IDF). TF-IDF merupakan sebuah teknik yang dapat digunakan didalam melakukan proses Transformasi Data dengan langkah-langkah dasar seperti yang telah dijelaskan pada sub-bab TF dan sub-bab DF. TF-IDF akan melakukan proses Transformasi Data dengan cara mengindeks *term* berdasarkan tingkat kepentingan masing-masing *term* [86]. Pada proses ini, yang terjadi adalah penentuan nilai bobot yang dilakukan oleh TF-IDF terhadap data *term* didalam *document*. Penentuan nilai bobot ini dilakukan secara seimbang dan mengikuti nilai / *value* dari TF dan DF. Namun, sesuai dengan nama teknik ini yang mengandung kata *invers* yang

dimaksudkan untuk melakukan *invers* terhadap nilai DF. *Invers* terhadap nilai DF arti sederhananya adalah jika terdapat sebuah kata yang memiliki nilai DF yang kecil (artinya hanya terdapat didalam sedikit document), maka akan dibuat menjadi nilai / *value* yang besar [87]. Hal ini bertujuan agar tercapainya nilai bobot yang relevan dan berimbang.

Dari proses ini, maka akan didapatkan output berupa nilai bobot berbentuk angka dengan rentang nol (0) hingga satu (1). Jika diperhatikan kembali dari penjelasan diatas, maka dapat dilihat bentuk transformasi data yang sebelumnya berbentuk teks berubah menjadi angka dengan cara mencari nilai bobot berdasarkan nilai / *value* TF dan nilai / *value* DF yang dilihat dari kemunculan kata / *term* didalam *document*. Perhitungan yang dilakukan TF-IDF dirumuskan didalam persamaan (2.2.).

$$W_{ij} = tf_{ij} \cdot idf_j$$
$$W_{ij} = tf_{ij} \cdot \log\left(\frac{D}{df_j}\right) \quad (2.2.)$$

W_{ij} merupakan bobot dari *term* (t_j) terhadap dokumen (d_i) dan tf_{ij} merupakan jumlah kemunculan dari *term* (t_j) dalam dokumen (d_i). D merupakan jumlah dari semua dokumen yang tersimpan dalam library dan df_j adalah jumlah dokumen yang mengandung *term* (t_j).

2.2.3. Reduksi Fitur

Reduksi fitur adalah tahapan lebih lanjut setelah proses *Data Cleaning* dan *Data Transformation*. Setelah data dibersihkan (*cleaning*) dan dirubah bentuknya (*transformation*), maka selanjutnya adalah pereduksian ukurannya. Proses yang terjadi pada *Data Reduction* adalah perubahan ukuran (size) data yang semula besar menjadi lebih kecil (direduksi). Proses reduksi data ini bertujuan untuk mengurangi komputasi yang berat pada mesin. Ukuran data yang besar akan membuat pengolahan data yang berat pada mesin dan (jika mesin mampu mengolah) secara otomatis akan memakan waktu yang lebih lama. Proses yang terjadi menjadi tidak efisien dengan banyaknya *space* mesin yang digunakan dan waktu pemrosesan yang lama. Dengan data yang

lebih kecil ukurannya (direduksi), maka proses yang dilakukan akan menjadi lebih efisien dengan waktu yang lebih cepat. Inilah sebab mengapa *Data Reduction* menjadi sangat penting pada *Text Processing*. Dalam banyak penelitian yang telah dilakukan oleh peneliti-peneliti didunia, terdapat peralatan (tools) yang biasa dipakai agar data dapat direduksi yaitu *Principal Component Analysis* (PCA) [88].

2.2.3.1. *Principal Component Analysis* (PCA)

Pada tahun 1901 seorang peneliti Karl Pearson memperkenalkan PCA, fungsi dari PCA adalah agar dapat memberikan penjelasan mengenai bentuk varian (ukuran korelasi antara dua buah variabel yang sama) serta kovarian (ukuran korelasi antara dua atau lebih variabel acak) pada rangkaian variabel yang dilalui kombinasi linier. Dalam kata lain, akan dilakukan perubahan yang dilakukan oleh PCA pada variabel-variabel sebenarnya (variabel asli) yang saling memiliki hubungan menjadi variabel-variabel terbaru yang tidak memiliki hubungan dengan dilakukannya pengurangan (reduksi) pada variabel-variabel itu kemudian memiliki dimensi yang lebih sedikit namun tetap mewakilkan identitas asli dari keragaman variabel sebenarnya. Penggunaan PCA sering kali menjadi satu teknik agar data/dimensi dapat di reduksi. Alasan dari penggunaan PCA, antara lain :

- Pengurangan (Reduksi) Data

  Agar dapat menyingkat suatu informasi yang memiliki variabel sebenarnya (variabel asli) dalam jumlah besar ke suatu set yang lebih kecil dibandingkan dengan dimensi gabungan yang baru dan rasio eror/loss informasi yang sangat minimum maka digunakan PCA.

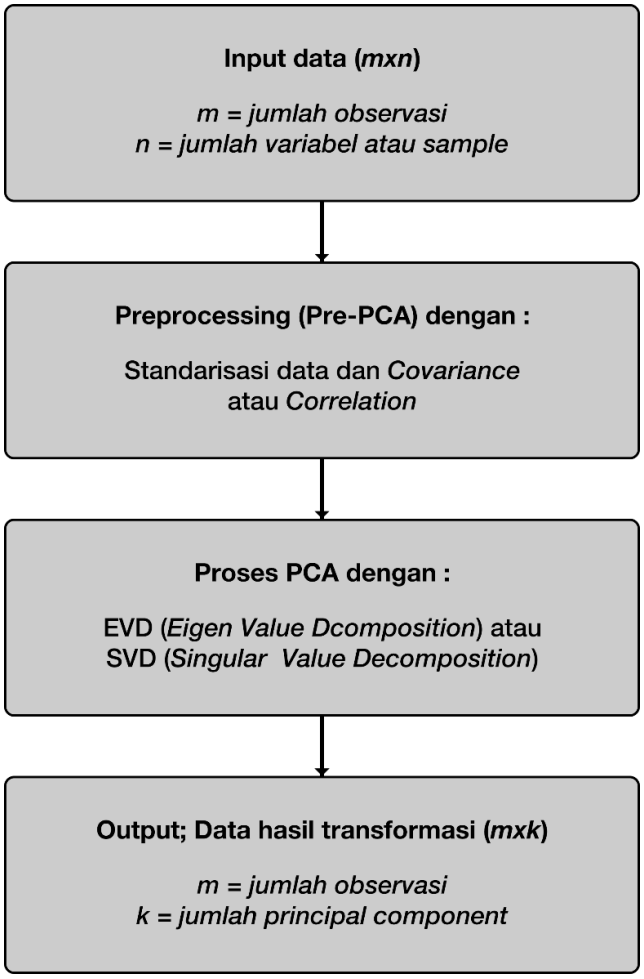
- Interpretasi

  Menemukan fitur-fitur penting di suatu dataset berukuran besar dapat menggunakan PCA. Hubungan antar data yang tidak diduga sebelumnya bisa diungkapkan oleh PCA, maka akan sangat mungkin timbulnya interpretasi hasil yang tidak biasa.

Agar analisis lebih efisien, penggunaan PCA merupakan langkah yang baik untuk dilakukan, analisis data dengan jumlah variabel input yang terlalu besar sangat baik dilakukan menggunakan PCA. Maka, cara kerja PCA antara lain:

- Kovarian matriks X dikalkulasikan dari data point
- Vektor *eigen* serta nilai *eigen* dikalkulasikan dengan sesuai
- Vektor *eigen* akan diurutkan menyesuaikan dengan nilai dalam urutan menurun
- Vektor k *eigen* pertama akan dipilih lalu dijadikan dimensi baru k
- Mentransformasi data point dimensi n asli menjadi dimensi k

Suatu bilangan skalar (hanya mempunyai nilai) atau disebut dengan nilai *eigen* serta sebuah matriks yang dapat mendefinisikan matriks A atau disebut pula vektor *eigen* (*eigenvector*), yang mana matriks A ialah suatu matriks persegi berukuran nxn. Detailnya, penggambaran langkah hirarkis dari PCA dapat dilihat di gambar 2.9.



Gambar 2.9. Langkah *principal component analysis* (PCA)

2.2.3.2. *Linear Discriminant Analysis* (LDA)

LDA merupakan algoritma lain yang terdapat didalam reduksi dimensi yang secara fungsi dan kegunaan sama dengan *Principle Component Analysis* (PCA). Kedua algoritma ini tujuannya adalah sama persis, yaitu untuk mengurangi dimensi data yang akan diolah didalam *classifier*. Akan tetapi, meskipun fungsi dan tujuannya sama, cara kerja dan logika algoritma yang digunakan dalam pengolahan data adalah pembeda diantara keduanya. LDA adalah hasil generalisasi dari teori *Discriminant* R. A. Fisher [89] yang pertama kali dikemukakan pada tahun 1936. Secara umum, LDA mampu melakukan berbagai macam pekerjaan didalam *Machine Learning* seperti *Face Recognition* [90][91], *Classification* [92][93], dan *Dimensionality Reduction* [94][95]. Namun, dua fungsi LDA yang paling populer adalah sebagai sebuah *classifier* dan sebagai sebuah

algoritma untuk mereduksi dimensi data. LDA dapat digunakan sebagai sebuah *classifier* dan melakukan klasifikasi dengan baik terhadap dataset yang bersifat *linear*. Walau demikian, bukan berarti LDA tidak mampu melakukan klasifikasi terhadap dataset yang tidak *linear*, hanya saja hasil yang didapatkan akan jauh lebih baik jika LDA digunakan untuk klasifikasi terhadap dataset yang *linear*. LDA didalam *Dimensionality Reduction* dapat dimanfaatkan untuk melakukan reduksi terhadap data fitur yang begitu besar dengan tujuan untuk mengurangi beban komputasi dan waktu pemrosesan. Secara teknis, logika yang digunakan LDA didalam melakukan reduksi fitur adalah dengan merubah jumlah fitur awal menjadi K-1, dimana K adalah jumlah (banyaknya) kelas. Jadi, seumpama sebuah dataset memiliki fitur sebanyak 100 fitur, dan didalam dataset tersebut terdapat 5 kelas, maka LDA akan mereduksi fitur tersebut menjadi 4 fitur saja (dari 100 fitur menjadi 4 fitur).

2.2.3.3. *Two-stage Feature Clustering* (TSFC)

TSFC adalah sebuah algoritma lain didalam reduksi dimensi yang fungsinya sama dengan *Principle Component Analysis* (PCA) dan *Linear Discriminant Analysis* (LDA), yaitu untuk melakukan reduksi terhadap data fitur yang akan dimasukkan ke dalam *classifier*. TSFC merupakan teknik reduksi data yang berdasar dari sebuah algoritma *clustering* yang populer, yaitu *Two Stage Clustering*. Proses reduksi fitur akan dapat dilakukan dengan memanfaatkan algoritma *clustering* ini. Sesuai dengan namanya, TSFC adalah teknik reduksi fitur yang memiliki dua tahap pemrosessan [96]. Pada tahap yang pertama (*first stage*), proses yang terjadi bertujuan untuk mendapatkan sebuah *sub-cluster* yang disebut dengan *cluster* inisial fitur (*initial feature cluster*). Pada tahap ini, kumpulan kata akan dipotong kedalam banyak *cluster* inisial fitur yang tujuannya adalah untuk mendapatkan fitur-fitur yang mirip dari sebuah set fitur yang kecil (*sub-cluster*). Dengan begitu, maka ukuran data fitur dapat diperkecil secara signifikan. Sedangkan pada tahap kedua yang didapatkan pada tahap kedua disebut dengan cluster fitur yang serupa (*similar feature cluster*). Dalam tahap ini, semua *sub-cluster* yang telah didapatkan pada tahap pertama akan dikoneksikan dan digabungkan antara satu sama lain secara langsung kedalam sebuah *cluster* yang baru. Metode ini mampu mendapatkan inti fitur dengan lebih baik daripada metode *clustering* biasa serta mengurangi hilangnya informasi-informasi penting dari fitur-fitur yang ada.

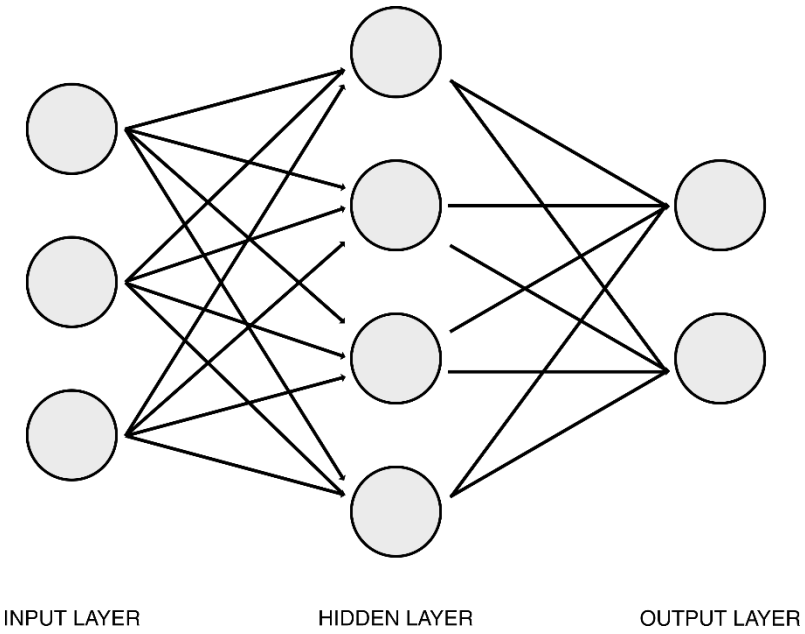
2.2.4. Penskalaan dan Normalisasi Fitur

2.3. *Machine Learning*

Machine learning merupakan bagian dari kecerdasan buatan. Tujuan dari *machine learning* secara umum adalah untuk memahami struktur dari data dan menyesuaikan data tersebut ke dalam model yang dapat dipahami dan digunakan oleh orang-orang. Terdapat beberapa pendekatan yang dapat digunakan dalam *machine learning*. Pendekatan *supervised* dan *unsupervised* adalah pendekatan yang mapan dan paling sering digunakan selain *Semi-supervised* dan *Reinforcement Learning*.

2.3.1. *Deep Neural Network* (DNN)

Deep Neural Network adalah sebuah model lanjut dari *Artificial Neural Network* (ANN). *Artificial Neural Network* (ANN) merupakan sebuah metode kecerdasan buatan/*Artificial Intelligence* (AI) yang dalam pembuatannya terinspirasi dari cara kerja sistem jaringan syaraf manusia. Itulah sebab mengapa model kecerdasan buatan ini diberi nama *Artificial Neural Network*. Sebuah *Neural Network* dibangun atas susunan ribuan neuron/node yang bersatu padu saling terhubung oleh sebuah *weight* (bobot). Neuron-neuron penyusun *Neural Network* terbagi/diletakkan ke dalam tiga jenis *Layer*, yaitu *Input Layer*, *Hidden Layer*, dan *Output Layer*. Didalam sebuah *layer*, setiap *neuron* akan bekerja melakukan komputasi secara paralel ketika melakukan proses pengolahan data yang keluarannya akan menjadi *input* neuron pada *layer* berikutnya. Oleh karena itu, pemrosesan antar *layer* didalam *Neural Network* adalah pemrosesan yang terurut dan seri, meskipun setiap neuronnya bekerja secara paralel dalam setiap *layer*. Secara sederhana, arsitektur ANN dapat dilihat pada gambar 2.10.



Gambar 2.10. Arsitektur *neural network*

Penjelasan dan urutan pemrosesan data yang dilakukan pada *neuron-neuron* didalam *Input Layer*, *Hidden Layer*, dan *Output Layer* pada ANN adalah:

1. *Input Layer* adalah lapisan pertama dimana semua data input/fitur dikelola untuk pertama kalinya. Proses ini akan menghasilkan output yang akan dijadikan sebagai input untuk neuron pada *Hidden Layer*.

2. Pada *Hidden Layer*, terjadi proses yang sama dengan sebelumnya (proses pada *input layer*) dengan data *input* yang diambil dari *output* proses *Input Layer*. Disini, data yang sudah berupa nilai tersebut diproses lagi hingga menghasilkan keluaran yang akan menjadi data input untuk *neuron* pada *Output Layer*.

3. Proses pada *Layer* yang terakhir (*Output Layer*) juga tidak jauh berbeda dengan *Layer* sebelumnya, yaitu proses komputasi *neuron* terhadap data masukan dari keluaran *Hidden Layer* hingga menghasilkan keluaran yang sebenarnya.

4. Hasil keluaran dari *Layer* terakhir beserta *loss* yang didapatkan akan dihitung berbalik arah hingga ke *Input Layer* (proses *Backward*) untuk melakukan *update weight* (bobot) demi

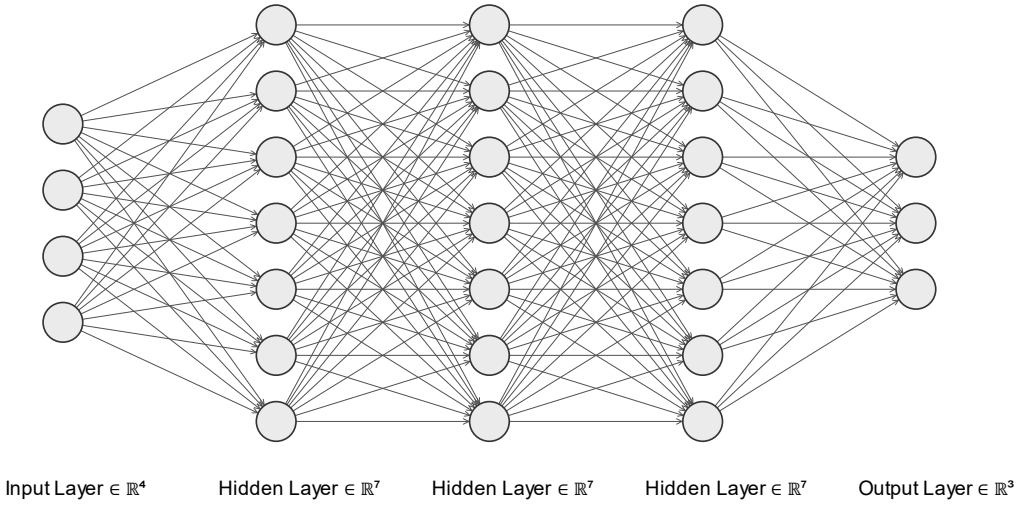
meningkatnya akurasi dan turunnya *loss*. Proses seperti itu akan terus berulang hingga nilai akurasi mencapai titik tertinggi dan nilai *loss* mencapai titik terendah (tergantung banyaknya *epoch* yang digunakan).

ANN mampu mengolah dan melakukan proses *learning* terhadap banyak bentuk data,baik itu data teks, data gambar, maupun data suara layaknya otak manusia. Hal ini tidak lepas dari sifat ANN yang adaptif yang membuat parameter-parameter ANN mudah dirubah sesuai keinginan kebutuhan penelitian [97]. Terdapat dua alasan yang membuktikan kinerja ANN yang menyerupai cara kerja otak manusia, adalah:

a. *Knowledge*, proses pembelajaran yang didapatkan dari lingkungannya akan disimpan dalam ANN.

b. *Synaptic Weights* (kekuatan hubungan antar neuron), keadaan lain yang digunakan agar hasil proses pembelajaran dapat disimpan.

Setelah *Artificial Neural Network*, model yang lebih kompleks dari itu adalah *Deep Neural Network* (DNN). Seperti yang telah dijelaskan pada paragraf pertama bahwa DNN adalah bentuk yang lebih rumit dari ANN. Secara teknis, perbedaan antara DNN dan ANN hanya terletak pada jumlah *Hidden Layer* yang digunakan. Dengan *Hidden Layer* yang lebih banyak, maka secara otomatis komputasi yang dilakukan pada DNN juga akan lebih kompleks. Komputasi yang lebih kompleks akan menyebabkan beban mesin bertambah. Namun, komputasi yang lebih kompleks adalah sebab dihasilkannya keluaran yang lebih baik dikarenakan DNN mampu mengolah data lebih *detail* dan lebih baik daripada ANN. Jalan pemrosesan data yang dilakukan DNN adalah sama dengan proses yang terjadi pada ANN. Hanya saja, karena banyaknya layer pemrosesan pada *Hidden Layer*, maka DNN mampu melakukan *learning* yang lebih baik terutama terhadap data/fitur yang invariant dan diskriminatif [98]. Tujuan dari dikembangkannya ANN menjadi DNN adalah untuk mengupayakan kemampuan *Neural Network* agar dapat memproses data yang lebih kompleks dengan baik, akurat, dan efisien. Tidak jauh berbeda dengan ANN, perhatikan arsitektur *Deep Neural Network* di Gambar 2.11.



Gambar 2.11. Arsitektur *deep neural network* (DNN)

Pada DNN, jumlah *Hidden Layer* dapat diatur sesuai kebutuhan hingga mendapatkan hasil yang optimal. Jumlah *Hidden Layer* yang bisa ditambah sesuai kebutuhan ini dapat dilihat pada gambar 2.11. Terlihat pada Gambar arsitektur tersebut jumlah *Hidden Layer* yang digunakan lebih dari satu. *Hidden Layer* yang berlapis ini akan mampu melakukan pemrosesan yang lebih baik karena terdapat perulangan proses yang banyak dan terus-menerus oleh setiap neuron yang terdapat dalam masing-masing *Hidden Layer* tersebut. Secara matematis, Setiap neuron tersebut melakukan proses komputasi dengan menggunakan persamaan (2.3) dan persamaan (2.4) [99].

Hidden Layer:

$$Z_{in_j}[j] = X_i[i] * V_{ij}[i,j] \tag{2.3.}$$

$$Z_{in\_j}[j] = Z_{in\_j}[j] + V_{ij}[i,j]$$

Output Layer:

$$Y_{ink}[k] = Z_z[j] * W_{jk}[j,k]$$

(2.4.)

$$Y_{ink}[k] = Y_{ink}[k] + W_{jk}[0,k]$$

Persamaan pertama pada persamaan (2.3) menunjukkan rumus untuk proses perhitungan yang dilakukan oleh *neuron-neuron* yang terdapat pada *Hidden Layer* untuk mendapatkan keluarannya ($Z_{in_j}[j]$). Keluaran ini akan didapatkan dengan mengalikan nilai input ($X_i[i]$) dengan nilai *weight* atau bobot ($V_{ij}[i,j]$). Persamaan yang kedua pada persamaan (2.3) adalah persamaan untuk proses *looping*/perulangan setelah didapatkan hasil pada proses yang dilakukan oleh persamaan pertama. Sedangkan untuk *output layer*, persamaan pertama yang ditunjukkan pada persamaan (2.4) adalah perhitungan yang dilakukan untuk menemukan keluaran/*output* ($Y_{ink}[k]$). Sama seperti pada rumus dan proses yang terjadi pada *Hidden Layer*, *Output* ini akan didapatkan dengan mengalikan nilai *output* dari *Hidden Layer* ($Z_z[j]$) dengan nilai *Weight*/Bobot. Persamaan kedua pada persamaan (2.4) juga sama seperti persamaan kedua pada persamaan (2.3), yaitu untuk rumus yang dilakukan untuk melakukan perulangan/*looping*.

Pada siklus *looping* (berulang) proses perhitungan akan terus berlangsung mencapai batas yang telah ditargetkan (*epoch*) dengan tujuan untuk membuat nilai *error*/*loss* menjadi serendah-rendahnya. Fungsi *Activation Function* dalam hal ini agar *neuron* dapat diaktifkan serta output yang dihasilkan menjadi non-linier, karena itu *Activation Function* berperan penting pada perhitungan di *Neural Network*.

Cara kerja dari *Activation Function* ialah nilai keluaran pada persamaan berdasarkan proses sebelumnya dikomparasi dengan nilai *threshold*. *Neuron* akan aktif, apabila hasil nilai keluaran perhitungan dari persamaan berdasarkan proses sebelumnya lebih dari nilai *threshold*. Adapun nilai *threshold* pada umumnya tidak semuanya sama, berbeda-beda sesuai dengan jenis *Activation Function* nya. Pada Tugas Akhir ini, *Activation Function* yang digunakan adalah *Rectified Linear Units* (ReLU) serta *Softmax*. *Input Layer* serta *Hidden Layer* akan menggunakan ReLU sebagai

Activation Function nya, sementara itu *Output Layer* nya menggunakan *Softmax* sebagai *Activation Function* nya. Fungsi dari *Softmax Activation Function* adalah agar nilai *error/loss* dapat dihitung menggunakan "*Cross Entropy*". Persamaan (2.5.) dan persamaan (2.6.) memperlihatkan rincian perhitungan matematis pada *Activation Function* ReLU serta *Activation Function Softmax*.

ReLU *Activation Function* :

$$f(relu) = \max\left(Z_{in_j}[j], 0\right) \tag{2.5.}$$

Softmax Activation Function :

$$Error = \frac{1}{N} * \sum_N \sum_I (target_{n,i} * \ln\left(Yk_{n,i}\right) \tag{2.6.}$$

$$delta_k[k] = Error\ Cross\ Entropy * f'(Y_{ink}[k])$$

Unit yang berperan dalam pengaktifan serta pe-nonaktifan *neuron* adalah ReLU, apabila keadaan *Hidden Layer* (Z_{in_j}) nilainya diatas dari nol maka akan terjadi pengaktifan *neuron*, kebalikannya untuk nilai *Hidden Layer* yang kurang dari nol atau nol maka *neuron* di non aktifkan. Berdasarkan dari fungsi ReLu sebagai *Activation Function*, maka bisa dilihat jika ReLU merupakan *Activation Function* yang paling sederhana serta proses pengolahan yang cepat agar dapat dipakai pada *Neural Network*, sementara itu *Softmax* yang berfungsi pada *Output Layer* untuk proses menghitung nilai *error/loss* pada *Neural Netwok* digunakan "*Cross Entropy*".

2.3.2. SVM

Support Vector Machine (SVM) adalah salah satu teknik didalam metode *Machine Learning* (ML) dengan pendekatan *Supervised Learning* yang fungsi dan kegunaannya sama dengan ANN dan DNN. SVM dan DNN adalah dua buah teknik yang sudah sangat sering

digunakan oleh banyak peneliti di dunia untuk melakukan klasifikasi, identifikasi, regresi, dan lain sebagainya yang mencakup pendekatan *supervised learning*. Meskipun fungsi dan kegunaannya sama, namun untuk algoritma pengolahan data yang digunakan antara SVM dan DNN sangat berbeda.

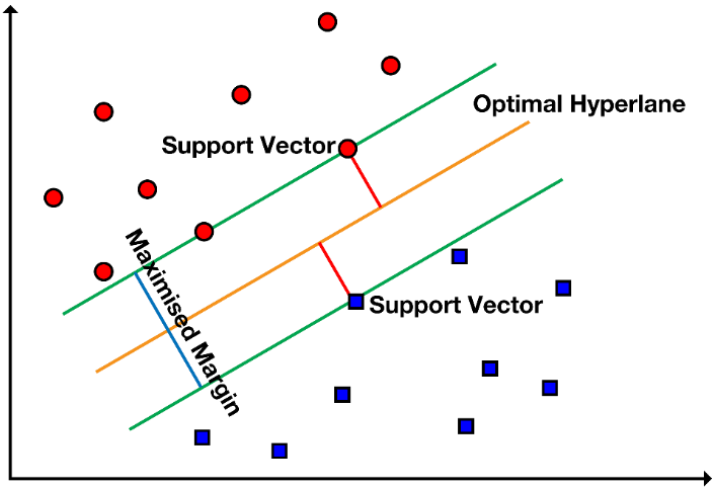
Pada tahun 1992, Vapnik, Boser, serta Guyon memperkenalkan SVM untuk pertama kalinya. Keunggulan pada SVM antara lain akurasi yang dihasilkan tinggi, mampu untuk melakukan pengolahan data yang memiliki dimensi tinggi, serta fleksibel mengikuti model dari berbagai sumber data. Keunggulan lainnya yang dimiliki SVM adalah memiliki target pada saat klasifikasi dilakukan secara optimal akan mendapatkan hasil akurasi setinggi-tingginya serta nilai *error/loss* yang di dapatkan serendah-rendahnya. Kinerja untuk memperoleh hasil akurasi tinggi serta rasio eror yang rendah membuat SVM bisa mendapatkan hasil yang optimal [100].

Fungsi-fungsi linier pada suatu fitur yang memiliki dimensi tinggi serta bantuan dari algoritma pembelajaran yang mengutamakan optimasi pada komputasinya dimanfaatkan oleh SVM. Kernel yang merupakan suatu teknik pada SVM yang akan melakukan transformasi data, setelah transformasi data, dicari batasan optimal antaran hasil keluaran yang memungkinkan oleh SVM. Dengan kata lain, akan dilakukan transformasi data yang sangat komplek oleh SVM, kemudian cara yang paling optimum agar data dapat dipisahkan sesuai dengan label atau keluaran yang telah ditetapkan akan dicari. Komputasi dilakukan pada *dot-product* (a, b) pada sebagian ruang fitur yang memiliki dimensi tinggi, dalam hal ini dilakukan oleh kernel. Ada kelebihan yang didapatkan berdasarkan hal tersebut, keterampilan dalam mendapatkan hasil batasan-batasan keputusan non-linier, serta fungsi kernel yang digunakan memberikan izin bagi penggunanya agar dapat diterapkan suatu *classifier* untuk data yang tidak mempunyai indikasi ruang vektor *fixed-dimensional* yang jelas.

Classifier yang memiliki fungsi untuk menemukan *hyperplane* (garis pemisah) yang paling baik agar vektor dapat dipisahkan dari kelas yang berbeda-beda ialah SVM. *Hyperplane* yang keberadaannya di posisi paling adil atau menengahi dua set objek yang berasal dari dua kelas yang tidak sama merupakan *hyperplane* yang paling baik. Berdasarkan penelitian Dendek dkk. [101], penentuan batasan pada *classifier* data linier,SVM mempunyai algoritma yang paling efisien. Banyak penelitian yang mengaplikasikan SVM misalnya Deteksi Wajah (Face Detection), Protein

Fold and Remote Homo-logy Detection, Klasifikasi Gambar (Classification of Images), Pengenalan Tulisan Tangan (Hand-Writing Recognition), Kontrol Prediktif Umum (Generalized Predictive Control (GPC)), Bioinformatics, Geo and Environmental Sciences, serta Kategorisasi Teks dan Hypertext (Text and Hypertext Categorization). Pada penelitian mengenai *Author Name Disambiguation* (AND), banyak peneliti yang menggunakan SVM sebagai metode *classifier*nya antara lain Giles dkk. [47], Dendek dkk. [101], dan Santana dkk. [33]

Dengan banyaknya penelitian pada SVM yang terus berkembang, hal ini membuat fungsi dari SVM semakin mendalam. Klasifikasi multi-class yang memiliki label lebih dari dua telah mampu dilakukan oleh SVM. Seiring dengan fungsi dari kernel, SVM mampu melakukan klasifikasi pada data kategorikal yang komplek. Cara kerja dari SVM dapat dilihat pada Gambar 2.12.



Gambar 2.12. *Support vector machine* (SVM)

Teori pembelajaran statistikal yang mendasari SVM untuk menjadi suatu teknik baru dalam melakukan klasifikasi serta regresi. Tidak seperti prinsip pada *Neural Networks*, penerapan prinsip *Structural Risk Minimization* (SRM) oleh SVM memiliki bukti lebih baik daripada *Neural Network* yang melakukan penerapan prinsip *Empirical Risk Minimization* (ERM). Walaupun baik SRM maupun ERM mempunyai keunggulan serta kelemahan tersendiri, batasan tertinggi pada *expected risk* dapat di minimalisir oleh SRM sementara itu kesalahan/eror pada training data dapat

di minimalisir oleh ERM [102]. Berdasarkan penjelasan sebelumnya, beragam macam kelas pada data akan ditransformasikan pada suatu bentuk *dot product*, yang berikutnya titik tengah *hyperplane* ditemukan yang memiliki batas dua atau lebih data tersebut pada suatu garis linier maupun non-linier. *Hyperplane* yang paling optimal akan dicari oleh SVM dari dua buah kategori data itu. Kinerja SVM secara sistematis dapat dilihat pada persamaan (2.7).

$$\beta 0 + \beta 1 * X1 + \beta 2 * X2 \dots \dots \beta n * Xn = 0 \quad (2.7.)$$

Kinerja dari SVM non-linier (data serta kategori yang lebih dari dua) secara sederhana diwakilkan oleh Persamaan (2.7.). nilai yang memiliki fungsi agar nilai optimal dari ditentukan dari *Hyperplane* disimbolkan dengan β serta nilai dari data masukan (input) disimbolkan dengan X. Perbedaan pada linier ialah jumlah seluruh data serta kategorinya, yang mana pada linier dua buah kategori hanya dibagi. Kinerja SVM secara linier apabila dirumuskan dapat dilihat pada persamaan (2.8.).

$$\beta 0 + \beta 1 * X1 + \beta 2 * X2 = 0 \quad (2.8.)$$

Hal yang digambarkan pada gambar 2.10. mewakilkan kinerja SVM linier seperti pada persamaan (2.7.). Pada suatu sampel data akan dilakukan pembatasan serta dilakukan pembedaan menjadi dua label ialah label positif (+1) serta label negatif (-1). Pembatas penentu pemberian label (*decision boundary)* inilah yang disebut dengan *Hyperplane.* Apabila *hyperplane* diwakili oleh $g(x) = \langle w, x \rangle + c$ sehingga untuk aturan menentukan pemberian label dalam matematis seperti persamaan (2.9.).

$$f(x) = \begin{cases} +1, jika\ g(x) \geq 1 \\ -1, jika\ g(x) \leq -1 \end{cases} \quad (2.9.)$$

Jarak yang berada antara titik positif serta negatif yang paling dekat dengan garis pembatas (*hyperplane*) disebut dengan *Maximised Margin*. Pada gambar 2.10., garis berwarna biru yang

mengilustrasikan margin memiliki arti jika terdapat jarak diantara garis kuning (*hyperplane*) dan garis hijau (titik batas positif dan negatif). Garis hijau yang tepat pada *dot product* disebut dengan *support vectors*. *Margin* yang mempunyai rentang lebih besar akan memberikan hasil performa pada saat klasifikasi yang lebih baik. Apabila label yang diolah jumlahnya dua maka proses akan lebih mudah. Mengenai hal tersebut yang menjadi pembeda yang paling jelas dari SVM linier serta SVM non-linier.

SVM mengimplementasikan teknik/fungsi pada suatu kernel dalam algoritmanya. Ruang yang memiliki input data dimensi rendah akan diambil oleh kernel serta melakukan transformasi menjadi ruang yang memiliki dimensi lebih tinggi. Dengan kata lain, konversi permasalahan yang tidak dapat dipisahkan (non-separable) dijadikan ke permasalahan yang dapat dipisahkan (separable) dengan ditambahkan dimensinya yang akan dilakukan oleh kernel. Tentu saja akan sangat bermanfaat untuk penyelesaian permasalahan non-linier. Dalam hal ini, kernel akan membantu SVM dalam membangun *classifier* agar mendapatkan nilai akurasi yang lebih tinggi.

Terdapat tiga buah kernel inti yang dimiliki SVM, pertama adalah *Linear Kernel*. Kernel paling dasar agar *dot product* dapat dibedakan ke dua buah kategori. Jumlah perkalian antar pasangan dari nilai masukan (input) atau disebut juga *Dot product* antara dua buah vektor. Persamaan dari fungsi *Linear Kernel* secara matematis dapat dilihat pada persamaan (2.10.).

$$K(x, xi) = sum(x * xi) \tag{2.10.}$$

Selanjutnya adalah *Polynomial Kernel*. *Polynomial Kernel* ialah bentuk yang lebih umum daripada *Linear Kernel*. Perbedaannya ialah *Polynomial Kernel* mampu untuk melihat perbedaan ruang input dengan melengkung atau secara non-linier, yang jelas berbeda dengan *Linear Kernel*. Fungsi *Polynomial Kernel* secara matematis dapat dilihat pada persamaan (2.11.).

$$K(x, xi) = 1 + \ sum(x * xi)^d \tag{2.11.}$$

d merupakan *degree* atau derajat yang memutuskan lekuk yang berasal dari ruang masukan (input) dalam *Polynomial Kernel*. Apabila *d = 1* memiliki arti sama saja dengan fungsi dari *Linear Kernel*. Penyesuaian perlu dilakukan oleh *degree* pada data, agar didapatkan klasifikasi yang akurat, selain itu agar didapatkan juga model yang efisien dengan klasifikasi dan data. Terakhir ialah *Radial Basis Function* (RBF), kernel yang paling sering digunakan dalam pen-klasifikasian menggunakan SVM. Keuntungannya, pemetaan ruang masukan (input) dapat RBF lakukan dengan dimensi yang tidak memiliki batas. Fungsi *RBF Kernel* secara matematis dapat dilihat pada persamaan (2.12.).

$$K(x, xi) = \exp\left(-gamma * sum(x - xi^2)\right) \quad (2.12.)$$

Suatu parameter yang memiliki rentang dari 0 hingga 1 disebut dengan *Gamma*. *Gamma* yang memiliki nilai terlalu tinggi memang sangat tepat untuk *training data* namun dapat mengakibatkan terjadinya *overfitting*. Nilai default yang paling baik untuk *gamma* adalah 0.1, namun tetap ada kemungkinan dapat dilakukan pengubahan yang sesuai dengan hasil yang diperoleh pada saat pengujian. Penentuan nilai *gamma* harus dilakukan secara manual pada algoritma pembelajaran SVM.

2.4. *Performance Measurements*

Suatu skema yang memiliki kegunaan agar dapat melakukan evaluasi serta pengukuran nilai suatu performa pada suatu model dan validasi dilakukan pada model tersebut merupakan pengertian dari *Performance Measurements*. Hal yang dapat diperoleh pada *Performance Measurements* ialah, skor Akurasi (*Accuracy)* di setiap kelas berdasarkan data yang digunakan pada proses klasifikasi, skor Sensitifitas (*Sensitivity*/*Recall)*, skor/nilai *F1-Score*, skor Spesifisitas (*Specificity)*, skor Presisi (*Precision)*, serta *Error* di setiap kelas dari data yang digunakan untuk klasifikasi yang dibantu oleh *Confusion Matrix*. Persamaan-persamaan untuk Rata-Rata Akurasi (*Average Accuracy)*, Presisi (*Precision*), *Recall*, serta *F1-Score* yang secara matematis dituliskan pada persamaan (2.13), (2.14.), (2.15.), serta (2.16.).

$$Average\ Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + tp_i + fp_i + tn_i}}{l} \quad (2.13.)$$

$$Precission_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \quad (2.14.)$$

$$Recall_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \quad (2.15.)$$

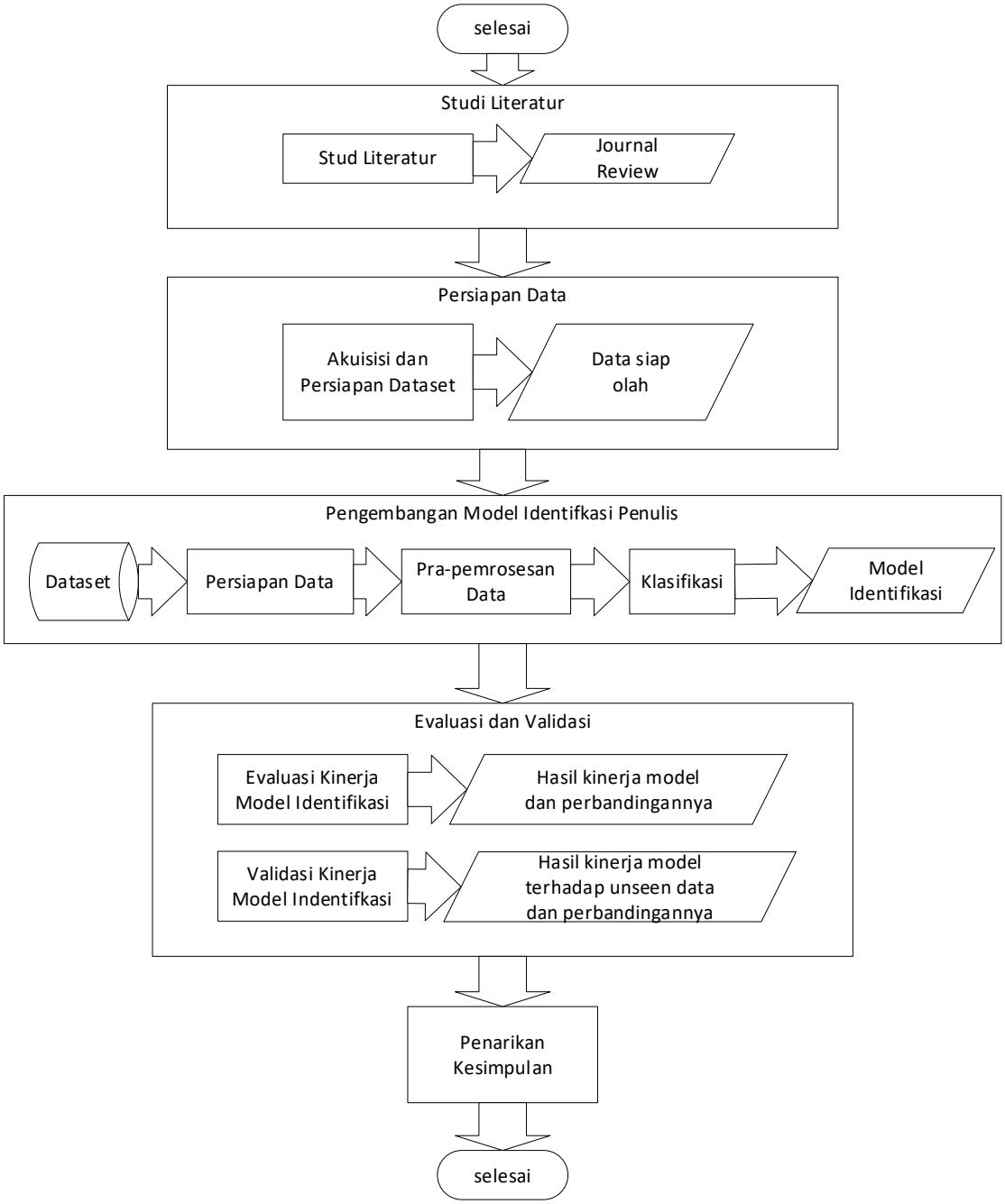
$$Fscore_M = \frac{(\beta^2 + 1) Precission_M Recall_M}{\beta^2 Precission_M + Recall_M} \quad (2.16.)$$

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dibahas metodologi pengembangan model identifikasi penulis pada data bibiliografi dengan menggunakan Teknik *Machine Learning* untuk meningkatkan akurasi dan kekokohan . Setiap tahap akan dijelaskan secara terperinci meliputi persiapan data, pra pengolahan data, klasifikasi, validasi model, analisis dan penarikan kesimpulan. Setiap langkah merupakan satu kesatuan kerangka penelitian yang akan dibahas setiap langkahnya.

3.1. Kerangka Penelitian

Pengembangan model identifikasi author pada data bibliografi terdiri dari empat fase utama yang tertuang dalam sebuah kerangka penelitian (gambar 3.1). Fase tersebut terdiri dari studi literatur, fase persiapan dataset bibliografi, fase pengembangan model identifkasi penulis, fase evaluasi dan validasi model, dan penarikan kesimpulan.



Gambar 3.1. Kerangka penelitian pengembangan model identifkasi penulis

3.2.Studi Literatur

Studi literatur dilakukan dengan menelusuri sumber-sumber tulisan yang pernah melakukan penelitian sebelumnya untuk mencari apa yang telah dikerjakan pada topik AND.

Penelusuran dilakukan pada jurnal-jurnal bereputasi selanjutnya diorganisasikan berdasarkan kronologi dan tematik.

3.3. Persiapan Dataset

Berbagai dataset digunakan dalam penelitian AND. Tiga dataset yang sering digunakan dan menjadi sering manjadi patokan adalah DBLP, Arnetminer, dan BDBComp [40]. Penelitian ini meenggunakan dataset yang dikembangkan oleh Jinseok Kim [65], dimana datset ini didapat dari hasil *cleaning process* terhadap dataset berbasis DBLP milik C. Lee Giles [44] yang diberi nama dataset *The GILES* dengan tujuan untuk memperbaiki, meminimalisir kekeliruan data dan menyempurnakan dataset dengan menghilangkan data yang berulang pada dataset tersebut. Kekeliruan yang dimaksud adalah *co-authors* yang tidak ada dan atau belum lengkap, tahun yang tidak sesuai, data yang berulang [103].

Dataset yang telah dibersihkan disinkronisasikan lagi dengan database terbaru pada perpustakaan digital DBLP dengan membandingkan nama penulis, tahun, judul, dan venue, dengan tujuan mengkoreksi titik terjadinya eror, seperti tahun yang tidak sesuai atau nama *co-authors* yang tidak lengkap sehingga dataset hasil *cleaning process* ini mengalami perubahan dan penambahan data yang lebih lengkap dan terbaru. Secara garis besar proses pembersihan yang dilakukan oleh Kim berupa penghilangan data yang berulang, pengisian data yang masih kosong (*null*) dan atau belum lengkap, update dan sinkronisasi data terbaru terhadap data pada website DBLP, dan pengelompokan entitas data terhadap atributnya sehingga dataset menjadi lebih rapi, terstruktur, dan terorganisir dengan baik.

Dataset Kim memiliki total 5018 data dengan 456 author berbeda (tabel 3.1). Dataset ini memiliki jumlah dan banyak data yang sama untuk setiap atributnya. Setiap atribut mewakilkan suatu informasi data dari seorang author yang terdiri dari *Author Name*, *Unique Author ID*, *Paper ID*, *Author List*, *Year*, *Venue*, dan *Title*. *Author Name* berisikan informasi nama dari author utama yang didapatkan dari DBLP, *Unique Author ID* sebagai label dari seorang author utama, *Paper ID* sebagai label untuk paper, *Author List* atau bisa disebut juga sebagai *Co-Authors* ialah informasi yang berisi nama dari author juga yang juga ikut terlibat dalam penulisan suatu paper bersama author utama, *Year* sebagai tahun publikasi dari dokumen yang bersangkutan, dan *Venue* ialah nama tempat atau lokasi dari suatu paper tersebut dipublikasikan. *Author Name* dan *Author List*

berisikan informasi yang sama, tetapi keduanya merepresentasikan informasi yang berbeda, dimana *Author Name* mewakili nama dari author utama, dan *Author List* mewakili nama dari author lainnya yang terlibat dalam suatu dokumen publikasi tersebut.

Tabel 3.1. Deskripsi dataset Kim

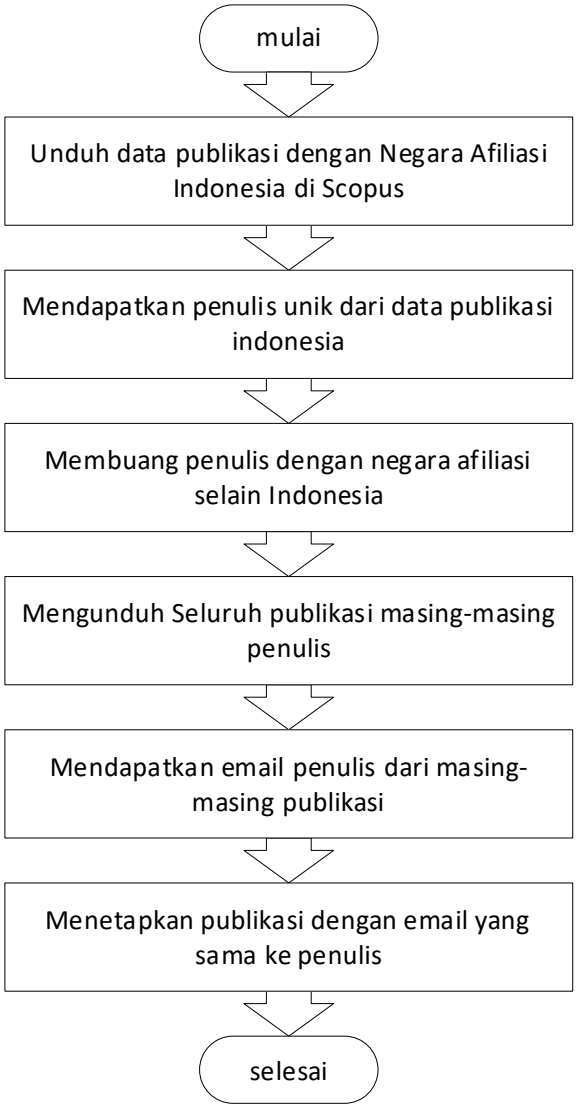
	Jumlah Data
Name instances	5018
Distinct authors	480
Distinct presented names	456
Distinct venues	1004
Distinct co-author names	4653
Year range	1959-2010
Synonym authors/row affected	46/1069
Homonym presented names/row affected	62/787
Non-synonym-homonym row affected	2988
Synonym-homonym row affected	174

Kinerja model juga akan diuji dengan dataset yang dikembangkan oleh penulis. Data set dikembangkan dengan beberapa tahap kegiatan (gambar 3.2) dengan data yang bersumber dari Scopus. Data publlikasi diunduh dari scopus dengan negara penulis Indonesia.

Dari data publikasi yang diunduh, diambil seluruh penulis yang mempunyai negara afiliasi Indonesia. Kemudian penulis-penulis tersebut diunduh publikasinya. Untuk memastikan penulis merupakan penulis Indonesia, dilakukan pemeriksaan negara afiliasi per publikasi. Penulis dengan negara afiliasi dominan Indonesia ditetapkan sebagai penulis Indonesia. Selanjutnya diunduh alamat email dari setiap penulis pada publikasi dari sumber yang beragam. Publikasi dengan alamat email yang sama dapat dipastikan merupkan publikasi penulis tersebut. Deskripsi hasil pengembangan dataset publikasi penulis Indonesia dapat dilihat pada tabel 3.2.

Tabel 3.2. Deskripsi dataset Indonesia

	Jumlah Data
Name instances	10212
Distinct authors	1484
Distinct presented names	1673
Distinct venues	1177
Distinct co-author names	9774
Year range	1995-2021
Synonym authors/row affected	203/1415
Homonym presented names/row affected	30/319
Non-synonym-homonym row affected	8422
Synonym-homonym row affected	56



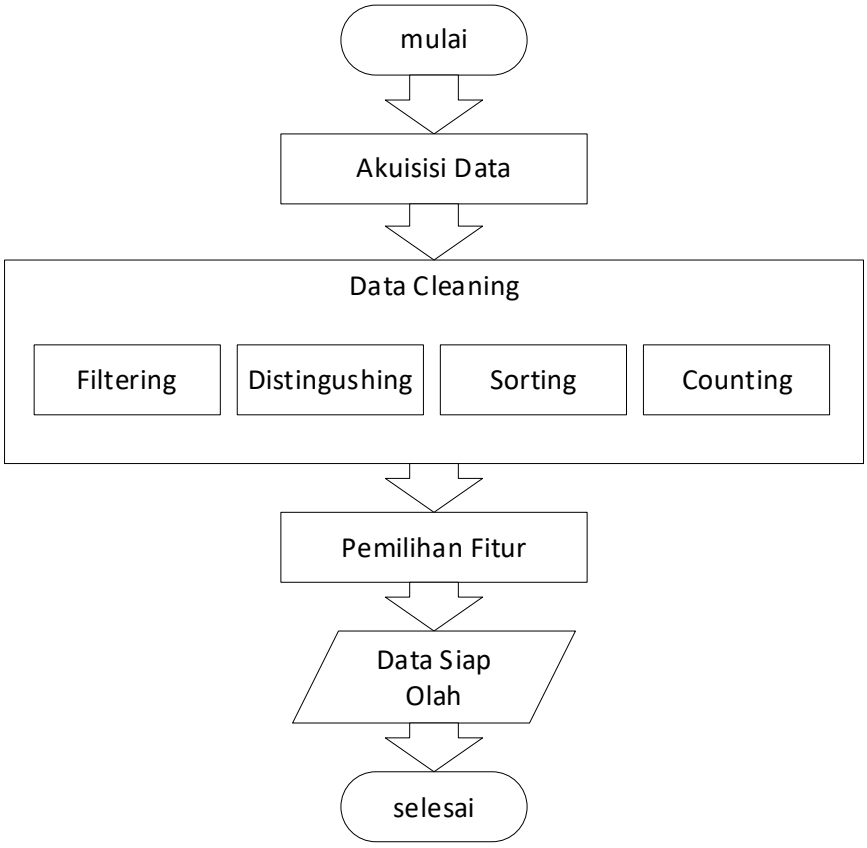
Gambar 3.2. Proses pembuatan dataset indonesia

3.4. Pengembangan Model Identifkasi Penulis

Pengembangan model indentifikasi penulis terdiri dari beberapa tahapan proses; persiapan data, pra-pemrosesan data dan pelatihan model dengan menggunakan data yang telah diproses. Hasil yang diharapkan berupa model identifkasi penulis yang memiliki akurasi yang baik dan kokoh.

3.4.1. Persiapan Data

Proses persiapan data terdiri dari beberapa tahapan seperti terlihat pada gambar 3.3.



Gambar 3.3. Proses persiapan data untuk identifikasi author

Dalam proses melakukan identifikasi author, langkah pertama yang harus dilakukan adalah akuisisi data untuk mendapatkan dataset yang akan digunakan sebagai sumber data dalam melakukan identifikasi author. Setelah proses akuisisi data dilakukan, langkah selanjutnya adalah melakukan proses pembersihan terhadap dataset. Proses pembersihan bertujuan untuk mendapatkan dataset yang baik dan efektif untuk digunakan dalam penelitian. Teknik yang dilakukan dalam proses pembersihan adalah *filtering* dan *distinction* terhadap dataset yang digunakan untuk membersihkan dataset dari data yang kosong (*null*) dan data yang berulang. Proses pembersihan yang dilakukan selanjutnya adalah *sorting* dan *counting*, kedua proses ini

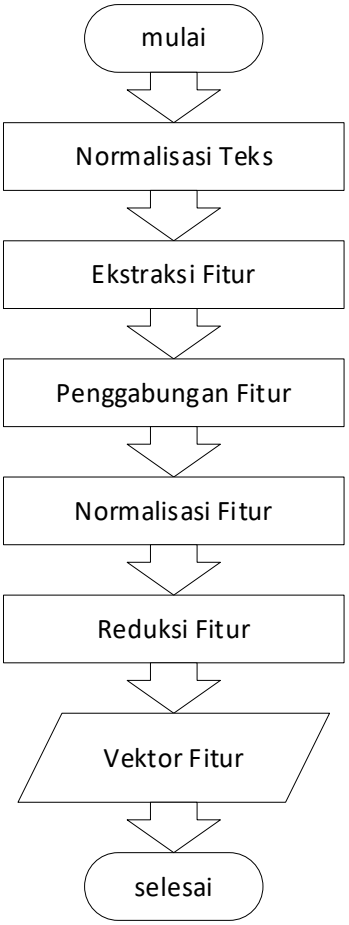
merupakan proses akhir dari proses pembersihan yang bertujuan untuk menghitung, memastikan dan melakukan pengecekan ulang terhadap dataset.

Langkah selanjutnya adalah proses pemilihan fitur yang memiliki peran paling berpengaruh terhadap variabel prediksi dalam dataset dan penelitian yang dilakukan. Dari total tujuh atribut yang terdapat pada dataset yang digunakan, enam atribut diantaranya dipilih dan akan digunakan karena memiliki potensi paling besar dalam melakukan identifikasi author. Keenam atribut tersebut adalah *Author Name*, *Unique Author ID*, *Author List*, *Title*, *Venue*, dan *Year*.

Enam buah atribut yang memiliki potensi dan dampak paling besar dalam penyelesaian permasalahan AND khususnya identifikasi author akan dibagi menjadi atribut fitur dan atribut label. Atribut fitur adalah atribut yang menjadi data input pada *classifier* yang digunakan. Atribut label adalah atribut yang menjadi acuan utama dari atribut fitur. Berbeda dengan atribut fitur yang digunakan sebagai input data, atribut label digunakan sebagai output data dari atribut fitur. Atribut label merupakan salah satu dari enam atribut yang memiliki data paling unik sehingga digunakan sebagai acuan utama untuk membedakan kelompok data yang diklasifikasikan. Untuk membedakan penulis digunakan atribut *Author ID*.

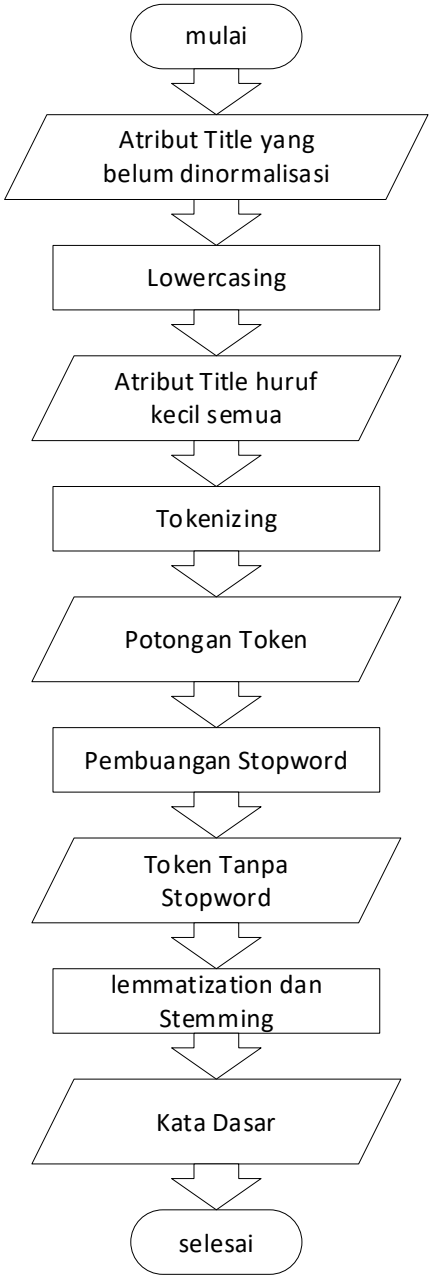
3.4.2. Pra-pemrosesan Data

Sebelum masuk ke tahap klasifikasi, perlu dilakukan pra pemrosesan data. pra pemrosesan data untuk identifikasi author terdiri dari lima tahap, normalisasi teks, ekstraksi fitur, penggabungan fitur dan normaliasi fitur. Kelima tahap ini menghasilkan vektor fitur yang akan menjadi masukan classifer (gambar 3.4.). dalam setiap tahap pra pemrosesan akan dicoba beberapa teknik untuk mendapatkan hasil terbaik.



Gambar 3.4. Pra-pemrosesan data untuk identifikasi author

Dari 5 atribut dan 1 label yang tersedia dari dataset, atribut *Title* merupakan sebuah frasa berbentuk teks. Teks ditransformasikan dalam beberapa cara untuk membuatnya konsisten. Beberapa teknik normalisasi teks digunakan dalam penelitian ini, lowercasing, tokenization, membersihkan *stopword*, dan normalisasi morfologi yang terdiri dari *lemmatization* dan *stemming* (Gambar 3.5). Lowercasing membuat *Title* yang terdiri dari campuran huruf kapital dan huruf kecil menjadi huruf kecil. *Tokenization* membuat *Title* menjadi potongan-potongan token. Sedangkan pembuangan *stopword* mendapatkan kata-kata penting dari token. Kata-kata dari token kemudian dicari kata dasarnya dengan menggunkan *lemmatization* dan *stemming*.



Gambar 3.5. Normalisasi atribut *title*

Setelah proses normalisasi teks pada atribut *title*, selanjutnya dilakukan proses ekstraksi fitur terhadap seluruh atribut termasuk attribut label. Sebelum masuk kedalam proses klasifikasi, teks tidak dapat langsung menjadi masukan *classifier,* karena masukan *classifer* harus berupa numerik. Untuk mengubah teks menjadi classifer diperlukan proses ekstraksi fitur. Pada penelitian ini metode ekstraksi yang diusulkan adalah *bag-of-words*[104] untuk seluruh attribut yang

berbentuk teks (tabel 3.3). Sebagai pembanding digunakan metode TF-IDF dan word2vec. Hasil dari proses ini adalah sebuah vektor dengan panjang tertentu dengan rentang nilai 0 sampai dengan 1 untuk masing-masing attribut. Untuk attribut *year* yang berbentuk numerik tidak dilakukan proses ekstraksi fitur. Tetapi fitur ini akan dinormalisasi dengan menggunakan teknik *MinMaxScaler* (persamaan 3.1) yang akan merubah data tahun menjadi bentuk numerik dalam format rentang angka yang berurutan antara 0 hingga 1. Tujuan normalisasi attribut *year* adalah untuk menyamakan rentang nilai dengan attribut yang lain agar peran setiap fitur menjadi berarti.

$$m = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

Dimana: M = nilai yang telah dinormalisasi
x = nilai fitur *year*
x_{min} = nilai maksimum pada fitur *year*
x_{max} = nilai minimum pada fitur *year*

Tabel 3.3. Metode ekstraksi fitur pada atribut-atribut bertipe teks

Attribut	**Metode Ektraksi Fitur**	
	Diusulkan	**Pembanding**
Author Name	*bag-of-words*	*TF-IDF, Word2vec*
Unique Author ID	*bag-of-words*	*TF-IDF, Word2vec*
Author List	*bag-of-words*	*TF-IDF, Word2vec*
Title	*bag-of-words*	*TF-IDF, Word2vec*
Venue	*bag-of-words*	*TF-IDF, Word2vec*

Setelah semua fitur selesai diolah dan ditransformasikan menjadi format numerik dan mempunyai rentang yang sama, selanjutnya semua fitur yang dihasilkan dari proses ekstraksi fitur dan normalisasi digabungkan kedalam sebuah vektor.

Vektor yang dihasilkan dari penggabungan fitur memiliki dimensi yang sangat besar. Hal tersebut mengakibatkan waktu komputasi yang lama dalam proses training dan testing identifikasi author. Untuk itu dibutuhkan sebuah proses reduksi fitur. Pada penelitian ini, metode reduksi fitur yang diusulkan adalah *two-stage feature clustering* (TSFC) [105]. Sejauh ini TSFC merupakan

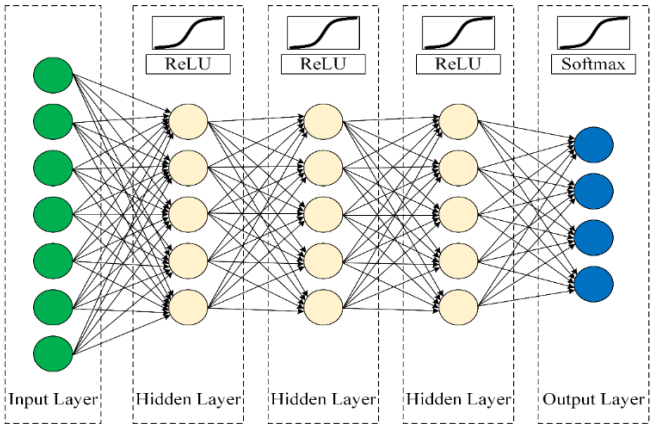
metode terbaik untuk reduksi fitur untuk klasifikasi teks pendek. Sebagai pembanding akan digunakan metode reduksi fitur PCA dan LDA. Hasil akhir dari keseluruhan pra pemrosesan menghasilkan sebuah *vector* yang siap untuk masuk kedalam *classifer*.

3.4.3. Klasifikasi

Fitur yang didapatkan dari proses pra pemrosesan data menjadi input data *classifier*. *Classifier* akan melakukan pembelajaran terhadap fitur-fitur yang diinput untuk menentukan sebuah referensi yang tertuju kepada author tertentu. Ada empat buah *classifier* yang dicoba untuk melakukan proses klasifikasi yaitu *Deep Neural Network* sebagai classififer yang diusulkan, *Support Vector Machine*, *Random Forest* dan *Naïve Bayes* sebagai pembanding.

a. Klasifikasi *Deep Neural Network* (DNN)

DNN merupakan pengembangan dari ANN yang dikembangkan dengan tujuan untuk bisa menyelesaikan pemrosesan data yang lebih sulit dan kompleks. DNN memiliki karakteristik dengan *Hidden Layer* lebih dari sama dengan dua yang menjadikan DNN bisa lebih banyak mempelajari fitur-fitur yang invarian dan bersifat diskriminatif [98]. Kemampuan ini dimungkinkan dengan meningkatkan jumlah *layer* dan *neuron* dalam setiap *layer*. Gambar 3.6. menampilkan arsitektur *classifier* DNN yang digunakan.



Gambar 3.6. Arsitektur DNN yang diusulkan

Dalam proses untuk mengidentifikasi author, klasifikasi DNN menggunakan *categorical cross-entropy* sebagai fungsi loss dan *Rectified Linear Unit* (ReLU) sebagai fungsi aktivasinya. Sebagai langkah untuk mencapai performa klasifikasi paling optimal, maka parameter tuning yang digunakan untuk klasifikasi DNN adalah *Hidden Layer* dengan 50, 100, 150, dan 200 neuron, tanpa *dropout*, dan dengan *dropout* 0.1 hingga 0.5. Parameter tuning selanjutnya yang sudah ditetapkan untuk semua skenario percobaan adalah pengaturan *epochs* bernilai 100, *learning rate* bernilai 0.01, dan *batch size* bernilai 64. Detail dari arsitektur DNN yang dibangun dan parameter tuning klasifikasi DNN dijelaskan oleh tabel 3.4.

Tabel 3.4. Arsitektur DNN dan parameter tuning

Layer	**Jumlah *Neuron***	**Fungsi Aktifasi**
Input	Hasil dari reduksi fitur	-
Hidden layer 1	50, 100, 150, 200	ReLU
Hidden layer 2	50, 100, 150, 200	ReLU
Hidden layer 3	50, 100, 150, 200	ReLU
Output layer	266	*Softmax*

b. *Support Vector Machine* (SVM)

SVM menjadi *classifier* kedua untuk melakukan klasifikasi. SVM merupakan sebuah *classifier* yang menghitung batas (*boundary*) dari suatu data untuk dipisahkan menjadi dua buah kelas, sederhananya SVM adalah *classifier* untuk dua kelas yang bisa dipisahkan secara linear. Akan tetapi, SVM membutuhkan fungsi kernel untuk memisahkan data yang memiliki banyak kelas atau disebut non-linear data. Fungsi kernel ini akan membantu pemetaan pola data ke beberapa *dimensional spaces*. Untuk mencapai hasil performa klasifikasi terbaik, *classifier* SVM diuji coba dengan menggunakan beberapa fungsi kernel yaitu *Linear Kernel Function, Polynomial Kernel Function*, *Radial Basis Function* (RBF), dan *Sigmoid Kernel Function*. Dengan fungsi kernel ini tadi, SVM sebagai *classifier* untuk dua kelas berubah menjadi *classifier* untuk multi-class (>2). Dalam penelitian ini, teknik *One-Against-One* (OAO) diterapkan untuk klasifikasi multi-class. Parameter lain yang akan di tuning adalah gamma dan C. Untuk parameter gamma dan C digunkan metode *Gridseaarch* untuk mencari nilai terbaik dari masing-masing parameter.

c. *Random Forest*

Classifier lain yang digunakan sebagai pembanding adalah *Random Forest*. *Random Forest* berkerja dengan membangun beberapa pohon keputusan pada saat pelatihan data dan menghasilkan kelas yang merupakan mode dari kelas pohon individu. *Random Forest* memiliki beberapa *hyperparameter* yang dapat di atur. *Hyperparameter* tersebut adalah jumlah pohon didalam hutan keputusan(*n_estimators)*, kedalaman setiap pohon (*max_depth*), jumlah sampel minimum yang diperlukan untuk membelah simpul daun internal (*min_samples_split*), jumlah sampel minimum yang diperlukan berada pada sebuah simpul daun (*min_samples_leaf*). Pada penelitian ini akan digunakan metode *exhaustive grid search* untuk tuning *hyperparameter random forest* [106].

d. *Naïve Bayes*

Classifer pembanding lain yang digunakan adalah *Naïve Bayes*. Pada penelitian ini digunakan *Multinomial Naïve Bayes* karena pada penelitian ini class yang digunakan lebih dari dua. Untuk *hyperparamter* alpha, akan ditentukan nilai optimumnya dengan *grid search* dengan mencoba nilai parameter yang mungkin.

3.5. Evaluasi dan Validasi Model

Dataset yang digunakan dalam penelitian ini bersumber dari dataset milik peneliti Dr. Giles (dataset *The GILES*) yang sudah dilakukan *Cleaning Process* oleh peneliti Jinseok Kim (dataset *DBLP Labeled Data*). *Cleaning Process* yang dilakukan oleh Jinseok Kim menghasilkan 5018 data dengan 480 author (*author labels*) yang berbeda dan 456 nama author yang berbeda, artinya untuk setiap nama author merujuk kepada author label (1 -480 author labels). Dataset yang digunakan terdiri dari *Author's Presented Name*, *Author Labels*, *List of Author Names*, *Venue*, dan *Title*. Dalam penelitian yang dilakukan, data author dengan referensi kurang dari 5 akan dibuat menjadi satu kelas lalu dihilangkan dari dataset sehingga total dataset berkurang dari 5018 menjadi 4419 data. Tabel 3.5. menjelaskan secara rinci perbandingan dataset mentah dengan dataset yang digunakan pada penelitian.

Tabel 3.5. Perbandingan komposisi dataset asli dan dataset yang digunakan

	Dataset Mentah	Dataset Digunakan
Jumlah data	5018	4419
Jumlah author	480	266
Jumlah nama tertulis	456	303
Jumlah *venue*	1004	923
Jumlah nama co-author tertulis	4653	3733
Rentang tahun	1959-2010	1959-2010
Synonym author sinonim/jumlah baris terdampak	46/1069	46/1120
Nama homonim/ jumlah baris terdampak	62/787	15/328
Jumlah baris terdampak non-homonim-sinonim	2988	2861
Jumlah baris terdampak homonim-sinonim	174	110

Dataset dibagi menjadi 80% untuk data training dan 20% untuk data testing. Berdasarkan jumlah referensi author yang ada pada dataset dan pembagian data untuk training dan testing, maka jumlah kelas yang digunakan dalam proses klasifikasi dapat ditentukan. Jumlah kelas yang digunakan dalam klasifikasi adalah 266 kelas author. Tabel 3.6 menjelaskan detail komposisi pembagian data untuk *training* dan *testing*.

Tabel 3.6. Komposisi data *training* dan data *testing*

	Raw dataset		***Prepared dataset***		***Training dataset***		***Testing dataset***	
	Record number	(%)	*Record number*	(%)	*Record number*	(%)	*Record number*	(%)
Synonym	1069	21.3	1120	25.3	900	25.5	220	24.9
Homonym	787	15.7	328	7.4	262	7.4	66	7.5
Synonym-Homonym	174	3.5	109	2.5	85	2.4	25	2.8
Non-Synonym-Homonym	2988	59.5	2861	64.7	2288	64.7	572	64.8
Total	5018	100	4418	99.9	3535	100	883	100

Dalam tahapan evaluasi dan validasi hasil, maka digunakan beberapa metrik pengkuran untuk mengukur tingkat kinerja model yang terdiri dari; *Average Accuracy* (3.2), *Precision* (3.3), *Recall* (3.4), dan *F1-Score* (3.5).

$$Average\ Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + tp_i + fp_i + tn_i}}{l} \tag{3.2}$$

$$Precision_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \tag{3.3}$$

$$Recall_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \tag{3.4}$$

$$Fscore_M = \frac{(\beta^2 + 1) Precision_M Recall_M}{\beta^2 Precision_M + Recall_M} \tag{3.5}$$

Dimana tp adalah *True Positive*, tn adalah *True Negative*, fp adalah *False Positive*, fn adalah *False Negative*, dan l adalah jumlah kelas.

Pengukuran kinerja model juga dilakukan terhadap keseluruhan data dan juga terhadap data yang masuk kedalam masing-masing empat kategori permasalahan AND, homonim, sinonim, homonim-sinonim, dan non-homonim-sinonim.

BAB IV
HASIL PENELITIAN

4.1. Pendahuluan

Bagian ini akan menjelaskan hasil-hasil yang telah dicapai dalam penelitian yang telah dilakukan. Hasil yang didapat dapat menjadi baseline dalam perhitungan kinerja sistem klasifikasi penulis pada data bibiligrafi.

4.2. Pengolahan data

Pada tahap ini, dataset yang digunakan adalah dataset yang di dikembangkan oleh Dr. Giles research lab at the Pennsylvania State University [107][44] dan telah dibersihkan oleh Kim [108]. Proses pembersihan data menghasilkan 5018 baris data dengan 480 penulis yang berbeda dan 456 nama yang berbeda, dimana setiap penulis memiliki 1 sampai dengan 480 publikasi.

Tahap pertama dalam pengolahan data adalah pengurangan data. Pada penelitian ini, data penulis yang memiliki data publikasi kurang dari 5 dibuang, sehingga jumlah data berkurang dari 5018 menjadi 4419 baris data. Rincian perbandingan jumlah data data sebelum pengurangan dan setelah pengurangan dapat dilihat pada tabel 4.1.

Table 4.1. Perbandingan rincian jumlah dataset sebelum dan sesudah dikurangi

	Data mentah	**Data Setelah Dikurangi**
Jumlah Data	5018	4419
Jumlah Penulis Berbeda	480	266
Jumlah Nama Berbeda	456	303
Venue Berbeda	1004	923
Nama *Co-author* Berbeda	4653	3733
Rentang Tahun	1959-2010	1959-2010

Tahap selanjutnya adalah memberikan kategori permasalahan AND yang terdiri dari *homonym, synonym, homonym-synonym,* dan *non-homonym-synonym*. Hasil dari pengkategorisasian data dapat dilihat pada tabel 4.2.

Table 4.2. Komposisi empat permasalahan AND pada dataset

	Raw data	***Prepared data***	***Percentage***
Synonym authors/row affected	46/1069	46/1120	25.3%
Homonym presented names/row affected	62/787	15/328	7.4%
Non-synonym-homonym row affected	2988	2861	64.7%
Synonym-homonym row affected	174	110	2.5%

4.3. Ekstraksi Fitur

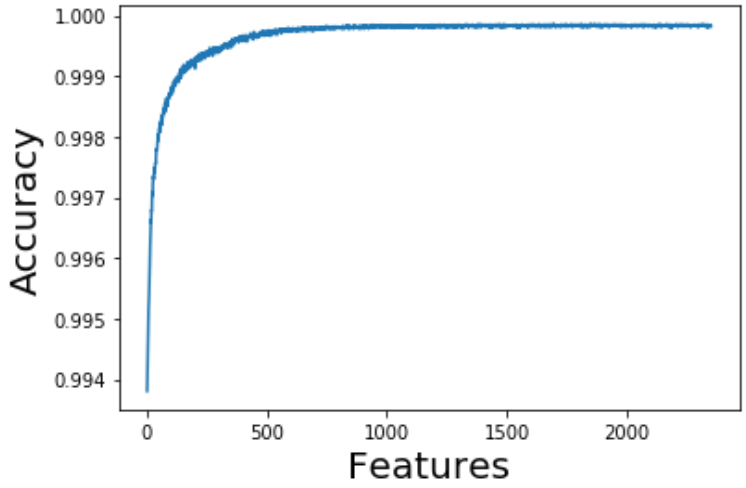
Masing-masing atribut pada dataset menghasilkan fitur sesuai dengan Teknik pembangkitan yang digunakan. Jumlah fitur yang dhasilkan sebanyak 7793. Detail fitur yang diasilkan masing-masing atribut dapat dilihat pada tabel 4.3.

Table 4.3. Jumlah Fitur yang dibangkitkan dari masing-masing atribut

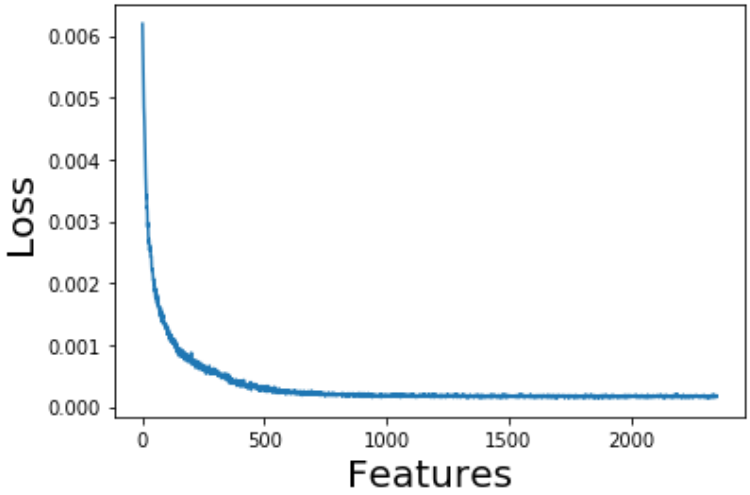
Attribute	***Feature Extraction Technique***	***Number of features***
Presented name	*Dummy variable*	303
List of author's names	*Dummy variable*	3733
Year	*None*	1
Venue	*Dummy variable*	923
Title	TF-IDF	2833
Total		7793

4.4. Reduksi Fitur

Fitur yang dihasilkan dari proses pembangkitan fitur menghasilkan jumlah fitur yang besar. Hal ini membutuhkan sumberdaya komputasi yang besar. untuk mengurangi jumlah fitur digunakan Teknik Principal Component Analysis. Gambar 4.3 menampilkan akurasi dan loss uji coba reduksi jumlah fitur. Uji coba dilakukan dengan jumlah fitur 2 sampai 2500 fitur. Dari hasil uji coba didapt fitur yang optimal sebanyak 1674 fitur.



(a)



(b)

Gambar 4.1. Kinerja PCA (a) akurasi, (b) losss

4.5. Evaluasi

Dataset dibagi menjadi dua bagian, 80% data latih dan 20% untuk data uji. Jumlah class yang digunakan pada klasifikasi ditentukan oleh jumlah keseluruhan author yaitu 266 class author. Tabel 4.4 menampilkan jumlah baris dan porsi dari masing-masing permasalahan AND yang dipengaruhi oleh proses pembersihan dan pemisahan data. Pembersihan data memberikan efek yang cukup kepada porsi dari permasalahan AND, sedangkan pemisahan data tidak terlalu memberikan efek yang signifikan.

Tabel 4.4. Komposisi data latih dan data uji

AND Problems	**Data Mentah**		**Data Disiapkan**		**Data Latih**		**Data Uji**	
	Jumlah Baris	**(%)**	**Jumlah Baris**	**(%)**	**Jumlah Baris**	**(%)**	**Jumlah Baris**	**(%)**
Synonym	1069	21.30%	1120	25.34%	900	25.46%	221	25%
Homonym	787	15.70%	328	7.42%	262	7.41%	66	7.46%
Synonym-Homonym	174	3.46%	110	2.50%	85	2.40%	25	2.83%
Non-Synonym-Homonym	2988	59.54%	2861	64.74%	2288	64.72%	572	64.70%
Total	5018	100.00%	4419	100.00%	3535	100.00%	884	100.00%

Pada penelitian, digunakan 4 besaran untuk mengevaluasi kinerja model, akurasi, presisi, recall dan F1-score. Untuk mendapatkan kinerja terbaik, beragam struktur DNN dicoba pada proses pembelajaran. Lima belas struktur ditentukan dan divalidasi untuk mencari model terbaik. Seluruh *classifier* di susun dalam dua proses, proses latih dan proses uji. Hasil uji coba dari 224 Struktur DNN dapat dilihat pada tabel 4.5.

Tabel 4.5. Akurasi model (%) dari 244 struktur DNN

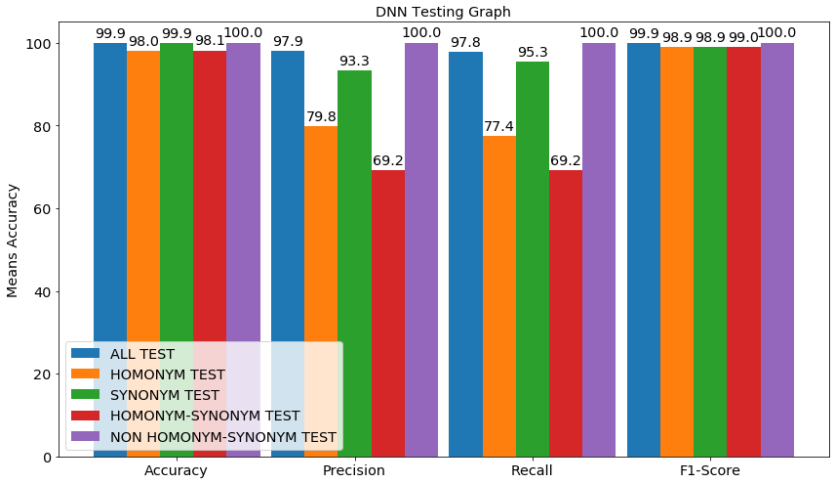
Neuron	*Learning Rate*	*Hidden Layer*							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
50	1E-01	99,5535	99,3068	99,2855	99,2660	99,2694	99,2660	99,2694	99,2651
50	1E-02	99,9804	99,9770	99,9481	99,9107	99,7891	99,7227	99,6538	99,5109
50	1E-03	99,9813	99,9796	99,9762	99,9702	99,9592	99,9294	99,9098	99,8622
50	1E-04	99,7967	99,8537	99,9192	99,9337	99,8852	99,8529	99,8316	99,7270
50	1E-05	99,3927	99,3102	99,2694	99,2838	99,2626	99,2847	99,2643	99,2728
50	1E-06	99,2524	99,2609	99,2524	99,2600	99,2583	99,2685	99,2668	99,2677
50	1E-07	99,2541	99,2498	99,2532	99,2507	99,2498	99,2515	99,2481	99,2498
100	1E-01	99,7355	99,3043	99,2694	99,2626	99,2541	99,2660	99,2694	99,2694
100	1E-02	99,9813	99,9779	99,9541	99,8461	99,7117	99,5926	99,4191	99,3808
100	1E-03	99,9830	99,9821	99,9779	99,9779	99,9660	99,9634	99,9541	99,9566
100	1E-04	99,9515	99,9762	99,9753	99,9736	99,9575	99,9524	99,9226	99,8707
100	1E-05	99,4718	99,2872	99,2889	99,2813	99,3238	99,2804	99,3485	99,2906
100	1E-06	99,2541	99,2498	99,2532	99,2566	99,2558	99,2634	99,2626	99,2634
100	1E-07	99,2498	99,2515	99,2498	99,2490	99,2498	99,2507	99,2515	99,2524
150	1E-01	99,7891	99,2958	99,2728	99,2694	99,2677	99,2694	99,2566	99,2558
150	1E-02	99,9770	99,9779	99,9439	99,7457	99,6079	99,4718	99,4335	99,3349
150	1E-03	99,9838	99,9821	99,9804	99,9762	99,9762	99,9660	99,9600	99,9396
150	1E-04	99,9753	99,9830	99,9804	99,9787	99,9770	99,9583	99,9405	99,9149
150	1E-05	99,5041	99,3230	99,3408	99,3646	99,3749	99,3689	99,3910	99,3570
150	1E-06	99,2600	99,2617	99,2566	99,2575	99,2575	99,2626	99,2634	99,2796
150	1E-07	99,2515	99,2507	99,2541	99,2566	99,2490	99,2507	99,2524	99,2498
200	1E-01	99,8180	99,2728	99,2694	99,2660	99,2660	99,2694	99,2694	99,2609
200	1E-02	99,9753	99,9779	99,9209	99,6036	99,5866	99,3706	99,3400	99,2762
200	1E-03	**99,9838**	99,9830	99,9804	99,9753	99,9745	99,9711	99,9643	99,9651
200	1E-04	99,9813	99,9813	99,9804	99,9813	99,9787	99,9719	99,9515	99,9132
200	1E-05	99,5620	99,3629	99,3689	99,3961	99,4148	99,4318	99,4888	99,4540
200	1E-06	99,2583	99,2626	99,2660	99,2753	99,2677	99,2719	99,2694	99,2677
200	1E-07	99,2524	99,2532	99,2583	99,2490	99,2498	99,2524	99,2507	99,2515

Hasil terbaik dihasilkan dari model struktur DNN dengan jumlah *hidden layer* 1, jumlah *neuron* 200 untuk setiap *layer* nya, dan nilai *dropout* sebesar 0.5. dari proses klasifikasi, model DNN diseleksi berdasarkan nilai akurasi, tetapi yang paling pentig adalah proses uji. Akurasi uji tertinggi untuk seluruh data adalah 99.99% untuk data latih dan 99.98 untuk data uji. Khusunya untuk nilai *recall* pada permasalahan *homonym-synonym* nilainya dibawah 70% (tabel 4.6) dikarenakan jumlah data sample yang memenuhi kondisi permasalahan *homonym-synonym* kurang dari kondisi yang lain yaitu sebanyak 110 data dari total sebanyak 4416 data atau sekitar 2.5% dari total seluruh data. Data yang tidak seimbang dapat menurunkan kinerja *classifier*.

Tabel 4.6. Kinerja klasifikasi DNN

AND Problem	***Accuracy***	***Precision***	***Recall***	***F1 Score***
All	99,9951	99,3289	99,3293	99,3123
	99,9838	97,9887	97,8641	99,9919
Homonym	99,2730	87,9282	88,9600	88,3163
	98,0303	79,8333	77,4524	98,9198
Synonym	100,0000	100,0000	100,0000	100,0000
	99,9380	93,3712	95,3463	99,9686
Homonym-Synonym	99,5848	84,1176	88,2353	85,6209
	98,1538	69,2308	69,2308	99,0769
Non-Homonym-Synonym	100,0000	100,0000	100,0000	100,0000
	100,0000	100,0000	100,0000	100,0000

Note : ■Training and ■Testing



Gambar 4.2. Kinerja DNN pada data uji

Untuk memvalidasi pendekatan dengan menggunkan DNN yang diusulkan, tiga *classifier*, *Naïve Bayes* ,*Random Forest* (RF), dan *Support Vector Machine* (SVM) dibandingkan dalam besaran akurasi, presisi, recall dan F1-Scroe. Tabel 4.7 menampilkan kinerja akurasi klasifikasi dari DNN dibandingkan dengan 3 classifer lainnya dengan nilai 99.98% untuk seluruh data, 98.03% untuk *homonym*, 99.38% untuk *synonym*, 98.15% untuk *homonym-synonym*, and 100% untuk *non-homonym-synonym*. Untuk keseluruhan besaran, presisi, *recall* dan F1-score, DNN juga memiliki nilai terbaik dibandingkan dengan *classifier* lainnya. Sebenarnya, dalam klasifikasi, SVM memberikan solusi unik, karena masalah optimalitasnya. Ini adalah keunggulan dibandingkan dengan DNN, yang memiliki beberapa solusi yang terkait dengan local minimum. Namun, DNN menggunakan struktur lapisan tersembunyi yang dalam, hal tersebut dapat mengatasi kelemahannya. Oleh karena itu, semua kinerja dapat ditingkatkan sekitar 1% dari SVM.

Tabel 4.7. Perbandingan kinerja DNN (%) dibandingkan dengan *classifier* lainnya

Classifier	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Naïve Bayes	99,8716	67,9285	73,1047	69,2674
Random Forest	99,8044	56,8699	58,2380	55,2470
SVM	99,9753	94,5677	95,2300	94,5997
DNN	99,9838	97,9887	97,8641	99,9919

Tabel 4.8. Perbandingan kinerja DNN (%) untuk setiap permasalahan AND dibandingkan dengan *classifier* lain

AND Problem	***Classifier***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Homonym	*Naïve Bayes*	97,2028	37,9487	43,4615	39,6410
	Random Forest	97,7553	42,4691	48,7302	44,0116
	SVM	98,3333	81,8333	80,8333	81,2778
	DNN	98,0303	79,8333	77,4524	98,9198
Synonym	*Naïve Bayes*	99,5059	70,0362	79,3536	73,3338
	Random Forest	99,4357	47,8859	55,9347	50,3384
	SVM	99,8918	94,2460	95,3953	94,1228
	DNN	99,9380	93,3712	95,3463	99,9686
Homonym-Synonym	*Naïve Bayes*	96,5714	55,3571	57,1429	56,1224
	Random Forest	94,9091	23,6364	31,8182	25,7576
	SVM	98,0000	75,0000	72,9167	73,8095
	DNN	98,1538	69,2308	69,2308	99,0769
Non-Homonym-	*Naïve Bayes*	99,8357	65,4443	70,3323	66,9863
Synonym	*Random Forest*	99,7518	48,6593	51,2205	48,3987
	SVM	99,9827	94,0594	95,0495	94,3894
	DNN	100,0000	100,0000	100,0000	100,0000

Metode dengan menggunkan *classifier* DNN yang diusulkan untuk identifkasi author pada data bibiliografi yang mengandung permasalahan *homonym* dan *synonym* menghasilkan kinerja yang baik, dengan melihat hasil percobaan, metode dengan DNN *classifier* memiliki kinerja yang baik dibandingkan dengan teknik lain. Dibandingkan dengan penelitan sebelumnya dengan dataset yang sama [108] metode yang diusulkan memiliki hasil yang baik pada besaran *recall*. Dapat kita lihat pada tabel 4.8, permasalahan *non-synonym-homonym* dapat dipecahkan secara sempurna oleh seluruh *classifier*, yaitu dengan akurasi sebesar 100%. Hal ini tidak lah mengejutkan, karena permasalahan *non-synonym-homonym* bukanlah permasalahan utama dalam identifikasi author. seperti yang sudah dijelaskan sebelumnya *synonym* dan *homonym* merupakan permasalahan yang paling kritis. Pada kenyataannya, permasalahan *homonym-synonym* adalah permasalahn yang sukar untuk diselesaikan dalam AND. Permasalahan ini harus di eksplorasi per permasalahan AND untuk identifkasi author berdasarkan karakteristiknya.

4.6. Pengujian Pada Dataset Publikasi Penulis Indonesia

Pengujian model terbaik DNN dilakukan pada dataset publikasi penulis Indonesia. Dengan menggunakan model terbaik DNN pada percobaan sebelumnya, dilakukan pra-pemrosesan dan penggunaan *hyperparameter* yang sama. Tahapan pra-pemrosesan menghasilkan sejumlah fitur untuk masing-masing atribut dengan total fitur sebanyak 22811 (tabel 4.9). Setelah dilakukan reduksi fitur dengan menggunakan PCA, total fitur yang dihasilkan sebanyak 6071.

Tabel 4.9. Jumlah fitur yang dihasilkan dari tahapan pra-pemrosesan dan reduksi fitur

Fitur	Jumlah fitur
Author(s) Name	1673
Label Author(s) (label)	1484
Co-Author(s)	9774
Venue	1177
Year	21
Title	10166
Total fitur (tanpa label)	22811
Total fitur (reduksi PCA dan tanpa label)	6071

Pengujian kinerja model pada dataset publikasi penulis Indonesia menghasilkan kinerja yang tidak jauh berbeda dengan penggunaan dataset Kim. Dapat dilihat pada tabel 4.10, secara keseluruhan untuk setiap permasalahan AND terjadi sedikit penurunan kinerja akurasi. Sedangkan untuk *precission* dan *recall* ada beberapa yang naik dan ada yang turun. Sedangkan untuk F1 score, secara keseluruhan terjadi penurunan kinerja. Hal ini memperlihatkan bahwa model yang dibangun cukup *robust* untuk mendeteksi author pada dataset yang berbeda.

Tabel 4.10. Perbandingan kinerja klasifikasi DNN pada data publikasi penulis Indonesia dengan dataset Kim

AND Problem s	***Accuracy***		***Precision***		***Recall***		***F1 Score***	
	Indonesia	**Kim**	**Indonesia**	**Kim**	**Indonesia**	**Kim**	**Indonesia**	**Kim**
All	100,00	99,99	100,00	99,33	100,00	99,33	100,00	99,31
	99,9953	99,98	96,0467	97,99	94,8978	97,86	95,0897	99,99
Homonym	100,00	99,27	100,00	87,93	100,00	88,96	100,00	88,31
	99,8231	98,03	94,3396	79,83	92,1384	77,45	92,9560	98,91
Synonym	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
	99,7940	99,94	75,5444	93,37	74,3382	95,35	73,2364	99,97
Homonym-Synonym	100,00	99,58	100,00	84,12	100,00	88,23	100,00	85,62
	94,8052	98,15	78,5714	69,23	78,5714	69,23	76,1905	99,08
Non-Homonym-Synonym	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
	99,9928	100,00	94,6122	100,00	93,8956	100,00	93,9006	100,00

Note : ■Training and ■Testing

BAB V
KESIMPULAN

6.1. Kesimpulan

Dari hasil percobaan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Metode yang diusulkan menghasilkan hasil yang baik untuk seluruh permasalahan AND dengan akurasi 99.98%.

2. Metode yang diusulkan memiliki kekohan yang baik dengan menghasilkan kinerja klasifikasi yang tidak jauh berbeda antar dataset

3. Metode yang diusulkan memecahkan permasalahan *synonym* lebih baik dari pada *homonym*.

4. Kinerja metode terkait dengan kombinasi permasalahn *homonym* dan *synomym* masih memiliki hasil yang tidak terlalu memuaskan.

5. Empat algoritma *machine learning* telah dibandingkan untuk mendapatkan kinerja yang presisi. Hasil percobaan menunjukan bahwa *Neural Network* dengan jumlah *layer* satu secara signifikan mengungguli teknik *machine learning* lainnya dengan akurasi sebesar 99.98%.

6. Dari hasil percobaan, DNN dan SVM menghasilkan kinerja yang paling baik pada permasalahan *synonym* dibandingkan *homonym*. Dengan teknik DNN yang diusulkan kinerja yang dihasilkan pada permasalahan *synonym* menghasilkan nilai akurasi, presisi, *recall* dan F1-score sebesar 99.94%, 93.37%, 95.35%, dan 99.97%.

6.2. Pengembangan Penelitian Lanjutan

Penelitian lanjutan yang dapat dikembangkan dari penlitian ini adalah pengembangan model identifikasi penulis dengan tujuan meningkatkan kinerja model untuk seluruh permasalahan AND dengan menerapkan rekayasa fitur berdasarkan pendekatan semantik untuk atribut judul.

DAFTAR PUSTAKA

[1] R. Hazra, A. Saha, S. B. Deb, and D. Mitra, "An efficient technique for author name disambiguation," *2016 IEEE Int. Conf. Curr. Trends Adv. Comput. ICCTAC 2016*, 2016.

[2] H. Han, C. Yao, Y. Fu, Y. Yu, Y. Zhang, and S. Xu, "Semantic fingerprints-based author name disambiguation in Chinese documents," *Scientometrics*, vol. 111, no. 3, pp. 1879–1896, 2017.

[3] I. Hussain and S. Asghar, "A survey of author name disambiguation techniques: 2010--2016," *Knowl. Eng. Rev.*, vol. 32, 2017.

[4] F. Momeni and P. Mayr, "Using Co-authorship Networks for Author Name Disambiguation," *Proc. 16th ACM/IEEE-CS Jt. Conf. Digit. Libr. - JCDL '16*, pp. 261–262, 2016.

[5] J. Schulz, "Using Monte Carlo simulations to assess the impact of author name disambiguation quality on different bibliometric analyses," *Scientometrics*, 2016.

[6] X. Lin, J. Zhu, Y. T. B, F. Yang, B. Peng, and W. Li, "A Novel Approach for Author Name," pp. 169–182, 2017.

[7] D. Han, S. Liu, Y. Hu, B. Wang, and Y. Sun, "ELM-based name disambiguation in bibliography," *World Wide Web*, vol. 18, no. 2, pp. 253–263, 2015.

[8] T. Huynh, K. Hoang, T. Do, and D. Huynh, "Vietnamese author name disambiguation for integrating publications from heterogeneous sources," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7802 LNAI, no. PART 1, pp. 226–235.

[9] N. Onodera *et al.*, "A method for eliminating articles by homonymous authors from the large number of articles retrieved by author search," *J. Assoc. Inf. Sci. Technol.*, vol. 62, no.

4, pp. 677–690, 2011.

[10] H. N. Tran, T. Huynh, and T. Do, “Author name disambiguation by using deep neural network,” in *Asian Conference on Intelligent Information and Database Systems*, 2014, pp. 123–132.

[11] J. Wang, K. Berzins, D. Hicks, J. Melkers, F. Xiao, and D. Pinheiro, “A boosted-trees method for name disambiguation,” *Scientometrics*, vol. 93, no. 2, pp. 391–411, 2012.

[12] A. P. de Carvalho, A. A. Ferreira, A. H. F. Laender, and M. A. Gonçalves, “Incremental unsupervised name disambiguation in cleaned digital libraries,” *J. Inf. Data Manag.*, vol. 2, no. 3, p. 289, 2011.

[13] Y. Liu, W. Li, Z. Huang, and Q. Fang, “A fast method based on multiple clustering for name disambiguation in bibliographic citations,” *J. Assoc. Inf. Sci. Technol.*, vol. 66, no. 3, pp. 634–644, 2015.

[14] C. Schulz, A. Mazloumian, A. M. Petersen, O. Penner, and D. Helbing, “Exploiting citation networks for large-scale author name disambiguation,” *EPJ Data Sci.*, vol. 3, no. 1, p. 11, 2014.

[15] J. Tang, A. C. M. Fong, B. Wang, and J. Zhang, “A unified probabilistic framework for name disambiguation in digital library,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 975–987, 2012.

[16] L. Tang and J. P. Walsh, “Bibliometric fingerprints: name disambiguation based on approximate structure equivalence of cognitive maps,” *Scientometrics*, vol. 84, no. 3, pp. 763–784, 2010.

[17] A. A. Ferreira, A. Veloso, M. A. Gonçalves, and A. H. F. Laender, “Self-Training Author Name Disambiguation for Information Scarce Scenarios,” vol. 65, no. 6, pp. 1257–1278, 2014.

[18] T. Gurney, E. Horlings, and P. Van Den Besselaar, “Author disambiguation using multi-aspect similarity indicators,” *Scientometrics*, vol. 91, no. 2, pp. 435–449, 2012.

[19] M. Imran, S. Z. H. Gillani, and M. Marchese, “A real-time heuristic-based unsupervised method for name disambiguation in digital libraries,” *D-Lib Mag.*, vol. 19, no. 9/10, 2013.

[20] M. Levin, S. Krawczyk, S. Bethard, and D. Jurafsky, “Citation-based bootstrapping for large-scale author disambiguation,” *J. Assoc. Inf. Sci. Technol.*, vol. 63, no. 5, pp. 1030–1047, 2012.

[21] G. Louppe, H. T. Al-Natsheh, M. Susik, and E. J. Maguire, “Ethnicity sensitive author disambiguation using semi-supervised learning,” in *International Conference on Knowledge Engineering and the Semantic Web*, 2016, pp. 272–287.

[22] H. Peng, C. Lu, W. Hsu, and J. Ho, “Disambiguating authors in citations on the web and authorship correlations,” *Expert Syst. Appl.*, vol. 39, no. 12, pp. 10521–10532, 2012.

[23] P. Wang, J. Zhao, K. Huang, and B. Xu, “A unified semi-supervised framework for author disambiguation in academic social network,” in *International Conference on Database and Expert Systems Applications*, 2014, pp. 1–16.

[24] J. Zhao, P. Wang, and K. Huang, “A semi-supervised approach for author disambiguation in KDD CUP 2013,” in *Proceedings of the 2013 KDD Cup 2013 Workshop*, 2013, p. 10.

[25] J. Zhu, Y. Yang, Q. Xie, L. Wang, and S.-U. Hassan, “Robust hybrid name disambiguation framework for large databases,” *Scientometrics*, vol. 98, no. 3, pp. 2255–2274, 2014.

[26] B.-W. On, I. Lee, and D. Lee, “Scalable clustering methods for the name disambiguation problem,” *Knowl. Inf. Syst.*, vol. 31, no. 1, p. 129, 2012.

[27] D. Shin, T. Kim, J. Choi, and J. Kim, “Author name disambiguation using a graph model with node splitting and merging based on bibliographic information,” *Scientometrics*, vol. 100, no. 1, pp. 15–50, 2014.

[28] X. Fan, J. Wang, X. Pu, L. Zhou, and B. Lv, “On graph-based name disambiguation,” *J. Data Inf. Qual.*, vol. 2, no. 2, p. 10, 2011.

[29] F. H. Levin and C. A. Heuser, “Evaluating the use of social networks in author name

disambiguation in digital libraries," *J. Inf. Data Manag.*, vol. 1, no. 2, p. 183, 2010.

[30] X. Wang, J. Tang, H. Cheng, and S. Y. Philip, "Adana: Active name disambiguation," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, 2011, pp. 794–803.

[31] Y. Liu and Y. Tang, "Network based Framework for Author Name Disambiguation Applications," *Int. J. u-and e-Service, Sci. Technol.*, vol. 8, no. 9, pp. 75–82, 2015.

[32] W.-S. Chin *et al.*, "Effective string processing and matching for author disambiguation," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3037–3064, 2014.

[33] A. F. Santana, M. A. Gonçalves, A. H. F. Laender, and A. A. Ferreira, "On the combination of domain-specific heuristics for author name disambiguation: the nearest cluster method," *Int. J. Digit. Libr.*, vol. 16, no. 3–4, pp. 229–246, 2015.

[34] X. Lin, J. Zhu, Y. Tang, F. Yang, B. Peng, and W. Li, "A novel approach for author name disambiguation using ranking confidence," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10179 LNCS, pp. 169–182, 2017.

[35] I. Hussain and S. Asghar, "LUCID: Author name disambiguation using graph Structural Clustering," in *2017 Intelligent Systems Conference, IntelliSys 2017*, 2018, vol. 2018-Janua, pp. 406–413.

[36] A. A. Ferreira and M. A. Gonçalves, "A Brief Survey of Automatic Methods for Author Name Disambiguation," vol. 41, no. 2, 2012.

[37] A. A. Ferreira, A. Veloso, M. A. Gonçalves, and A. H. F. Laender, "Effective self-training author name disambiguation in scholarly digital libraries," in *Proceedings of the 10th annual joint conference on Digital libraries*, 2010, pp. 39–48.

[38] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis, "Two supervised learning approaches for name disambiguation in author citations," in *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004.*, 2004, pp. 296–305.

[39] S. F. Schifano, T. Sgarbanti, and L. Tomassetti, "Authorship recognition and

disambiguation of scientific papers using a neural networks approach," in *Proceedings of Science*, 2018, vol. 327.

[40] S. S. Khan and M. G. Madden, "A survey of author name disambiguation techniques: 2010–2016," *Knowl. Eng. Rev.*, vol. 00, no. January, pp. 1–24, 2017.

[41] N. R. Smalheiser and V. I. Torvik, "Author name disambiguation," *Annu. Rev. Inf. Sci. Technol.*, vol. 43, no. 1, pp. 1–43, 2009.

[42] J. Kim, "Evaluating author name disambiguation for digital libraries: a case of DBLP," *Scientometrics*, vol. 116, no. 3, pp. 1867–1886, 2018.

[43] T. Backes, "Effective Unsupervised Author Disambiguation with Relative Frequencies," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2018, pp. 203–212.

[44] H. Han, W. Xu, H. Zha, and C. L. Giles, "A hierarchical naive Bayes mixture model for name disambiguation in author citations," in *Proceedings of the 2005 ACM symposium on Applied computing*, 2005, pp. 1065–1069.

[45] W. Liu *et al.*, "Author Name Disambiguation for PubMed," vol. 65, no. 4, pp. 765–781, 2014.

[46] G. Louppe, H. T. Al-natsheh, and M. S. B, "Using Semi-supervised Learning," vol. 3, pp. 272–287, 2016.

[47] L. Giles, "Two Supervised Learning Approaches for Name," pp. 296–305, 2004.

[48] B. Zhang and M. Al Hasan, "Name disambiguation in anonymized graphs using network embedding," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1239–1248.

[49] J. Zhu, X. Wu, X. Lin, C. Huang, G. P. C. Fung, and Y. Tang, "A novel multiple layers name disambiguation framework for digital libraries using dynamic clustering," *Scientometrics*, vol. 114, no. 3, pp. 781–794, 2018.

[50] C. Chen, J. Hu, and H. Wang, "Clustering technique in multi-document personal name

disambiguation," *ACL-IJCNLP 2009 - Jt. Conf. 47th Annu. Meet. Assoc. Comput. Linguist. 4th Int. Jt. Conf. Nat. Lang. Process. AFNLP, Proc. Conf.*, no. August, pp. 88–95, 2009.

[51] Z. Yu and B. Yang, "Researcher Name Disambiguation: Feature Learning and Affinity Propagation Clustering," in *International Symposium on Methodologies for Intelligent Systems*, 2018, pp. 225–235.

[52] J. J. Kim, J. J. Kim, and J. Owen-Smith, "Generating automatically labeled data for author name disambiguation: an iterative clustering method," *Scientometrics*, vol. 118, no. 1, pp. 253–280, 2019.

[53] B. Zhang, M. Dundar, and M. Al Hasan, "Bayesian non-exhaustive classification a case study: Online name disambiguation using temporal record streams," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 1341–1350.

[54] G. S. Mann and D. Yarowsky, "Unsupervised personal name disambiguation," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, 2003, pp. 33–40.

[55] D. Zhang, J. Tang, J. Li, and K. Wang, "A constraint-based probabilistic framework for name disambiguation," *Int. Conf. Inf. Knowl. Manag. Proc.*, pp. 1019–1022, 2007.

[56] M. Khabsa and C. L. Giles, "Online person name disambiguation with constraints categories and subject descriptors," *Proc. 15th ACM/IEEE-CE Jt. Conf. Digit. Libr. Knoxville, Tennessee, USA*, no. 2, pp. 37–46, 2015.

[57] V. I. Torvik, M. Weeber, D. R. Swanson, and N. R. Smalheiser, "A probabilistic similarity metric for Medline records: a model for author name disambiguation.," *AMIA Annu. Symp. Proc.*, vol. 56, no. 2, p. 1033, 2003.

[58] I. Hussain and S. Asghar, "Author Name Disambiguation by Exploiting Graph Structural Clustering and Hybrid Similarity," *Arab. J. Sci. Eng.*, vol. 43, no. 12, pp. 7421–7437, 2018.

[59] A. Veloso, A. A. Ferreira, M. A. Gonçalves, A. H. F. Laender, and W. M. Jr, "Cost-effective

on-demand associative author name disambiguation," *Inf. Process. Manag.*, vol. 48, no. 4, pp. 680–697, 2012.

[60] A. F. Santana, M. Andr, A. H. F. Laender, and A. A. Ferreira, "Incremental Author Name Disambiguation by Exploiting Domain-Specific Heuristics," vol. 00, no. 00, 2016.

[61] A. A. Ferreira, R. Silva, M. A. Gonçalves, A. Veloso, and A. H. F. Laender, "Active Associative Sampling for Author Name Disambiguation," pp. 175–184, 2012.

[62] P. Andruszkiewicz and S. Szepietowski, "Person Name Disambiguation for Building University Knowledge Base," no. Ii, pp. 270–279, 2016.

[63] K. Yang and Y. Wu, "Author Name Disambiguation in Citations," pp. 335–338, 2011.

[64] J. Liu, "Ranking-Based Name Matching for Author Disambiguation in Bibliographic Data."

[65] J. Kim, "Evaluating author name disambiguation for digital libraries : a case of DBLP," *Scientometrics*, vol. 116, no. 3, pp. 1867–1886, 2018.

[66] L. Reijnhoudt, R. Costas, E. Noyons, K. Börner, and A. Scharnhorst, "'Seed+ expand': a general methodology for detecting publication oeuvres of individual researchers," *Scientometrics*, vol. 101, no. 2, pp. 1403–1417, 2014.

[67] H. Kawashima and H. Tomizawa, "Accuracy evaluation of Scopus Author ID based on the largest funding database in Japan," *Scientometrics*, vol. 103, no. 3, pp. 1061–1071, 2015.

[68] M. J. Lerchenmueller and O. Sorenson, "Author disambiguation in PubMed: Evidence on the precision and recall of author-ity among NIH-funded scientists," *PLoS One*, vol. 11, no. 7, p. e0158731, 2016.

[69] T. Martin, B. Ball, B. Karrer, and M. E. J. Newman, "Coauthorship and citation patterns in the Physical Review," *Phys. Rev. E*, vol. 88, no. 1, p. 12814, 2013.

[70] R. Sinatra, D. Wang, P. Deville, C. Song, and A.-L. Barabási, "Quantifying the evolution of individual scientific impact," *Science (80-.).*, vol. 354, no. 6312, p. aaf5239, 2016.

[71] Y. Zhang, F. Zhang, P. Yao, and J. Tang, “Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop.,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1002–1011.

[72] K. Kim, A. Sefid, B. A. Weinberg, and C. L. Giles, “A Web Service for Author Name Disambiguation in Scholarly Databases,” in *Proceedings - 2018 IEEE International Conference on Web Services, ICWS 2018 - Part of the 2018 IEEE World Congress on Services*, 2018, pp. 265–273.

[73] V. I. Torvik and N. R. Smalheiser, “Author name disambiguation in MEDLINE,” *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 3, pp. 1–29, 2009.

[74] P. Treeratpituk and C. L. Giles, “Disambiguating Authors in Academic Publications using Random Forests Categories and Subject Descriptors,” *Med. Biochem.*, pp. 39–48, 2009.

[75] M. Khabsa, P. Treeratpituk, and C. L. Giles, “Large scale author name disambiguation in digital libraries,” *Proc. - 2014 IEEE Int. Conf. Big Data, IEEE Big Data 2014*, pp. 41–42, 2015.

[76] J. Tang, L. Yao, D. Zhang, and J. Zhang, “A combination approach to web user profiling,” *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 1, 2010.

[77] I. S. Kang, P. Kim, S. Lee, H. Jung, and B. J. You, “Construction of a large-scale test set for author disambiguation,” *Inf. Process. Manag.*, vol. 47, no. 3, pp. 452–465, May 2011.

[78] Y. Qian, Q. Zheng, T. Sakai, J. Ye, and J. Liu, “Dynamic author name disambiguation for growing digital libraries,” *Inf. Retr. J.*, vol. 18, no. 5, pp. 379–412, 2015.

[79] T. K. Saha, B. Zhang, and M. Al Hasan, “Name disambiguation from link data in a collaboration graph using temporal and topological features,” *Soc. Netw. Anal. Min.*, vol. 5, no. 1, pp. 1–14, 2015.

[80] V. Balakrishnan and E. Lloyd-yemoh, “Stemming and lemmatization: A comparison of retrieval performances,” pp. 174–179.

[81] L. Che, X. Yang, and L. Wang, “Text feature extraction based on stacked variational autoencoder,” *Microprocess. Microsyst.*, vol. 76, p. 103063, Jul. 2020.

[82] M. A. El Affendi and K. H. S. Al Rajhi, “Text encoding for deep learning neural networks: A reversible base 64 (Tetrasexagesimal) Integer Transformation (RIT64) alternative to one hot encoding with applications to Arabic morphology,” *6th Int. Conf. Digit. Information, Networking, Wirel. Commun. DINWC 2018*, vol. 64, pp. 70–74, 2018.

[83] J. D. Prusa and T. M. Khoshgoftaar, “Designing a better data representation for deep neural networks and text classification,” *Proc. - 2016 IEEE 17th Int. Conf. Inf. Reuse Integr. IRI 2016*, pp. 411–416, 2016.

[84] J. D. Prusa and T. M. Khoshgoftaar, “Improving deep neural network design with new text data representations,” *J. Big Data*, vol. 4, no. 1, 2017.

[85] N. Azam and J. Yao, “Comparison of term frequency and document frequency based feature selection metrics in text categorization,” *Expert Syst. Appl.*, vol. 39, no. 5, pp. 4760–4768, 2012.

[86] R. Goyena, “Climate Change 2013--The Physical Science Basis Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2019.

[87] L. Adhianto *et al.*, “HPCTOOLKIT: Tools for performance analysis of optimized parallel programs,” *Concurr. Comput. Pract. Exp.*, vol. 22, no. 6, pp. 685–701, 2010.

[88] B. Malley, D. Ramazzotti, and J. T. Wu, “Secondary Analysis of Electronic Health Records,” *Second. Anal. Electron. Heal. Rec.*, pp. 1–427, 2016.

[89] F. R. S. R. A. FISHER, Sc.D., “THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS,” vol. 1, no. 1, pp. 1–8, 1954.

[90] X. Wang and X. Tang, “Dual-space linear discriminant analysis for face recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, no. 3, pp. 0–5, 2004.

[91] T. K. Kim and J. Kittler, “Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 318–327, 2005.

[92] S. Pang, S. Ozawa, and N. Kasabov, “Incremental linear discriminant analysis for classification of data streams,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 35, no. 5, pp. 905–914, 2005.

[93] B. Zu, K. Xia, W. Du, Y. Li, A. Ali, and S. Chakraborty, “Classification of hyperspectral images with robust regularized block low-rank discriminant analysis,” *Remote Sens.*, vol. 10, no. 6, pp. 862–873, 2018.

[94] M. Zhao, Z. Zhang, T. W. S. Chow, and B. Li, “A general soft label based Linear Discriminant Analysis for semi-supervised dimensionality reduction,” *Neural Networks*, vol. 55, pp. 83–97, 2014.

[95] M. Sugiyama, “Local fisher discriminant analysis for supervised dimensionality reduction,” *ACM Int. Conf. Proceeding Ser.*, vol. 148, pp. 905–912, 2006.

[96] F. Li, Y. Yin, J. Shi, X. Mao, and R. Shi, “Method of Feature Reduction in Short Text Classification Based on Feature Clustering,” no. 2, pp. 1–13, 2019.

[97] S. Shanmuganathan, “Studies in Computational Intelligence 628 Artificial Neural Network Modelling,” pp. 145–159, 2016.

[98] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, “Feature Learning in Deep Neural Networks - Studies on Speech Recognition Tasks,” pp. 1–9, 2013.

[99] S. Nurmaini, R. U. P, M. N. R, and A. Gani, “Cardiac Arrhythmias Classification Using Deep Neural Networks and Principle Component Analysis Algorithm,” *Int. J. Adv. Soft Compu. Appl.*, vol. 10, no. 2, 2018.

[100] H. S. Ooi, G. Schneider, T. Lim, Y. Chan, B. Eisenhaber, and F. Eisenhaber, “Chapter 8 Biomolecular Pathway Databases,” *Methods Mol. Biol.*, no. January 2010, pp. 129–144, 2014.

[101] P. J. Dendek, Ł. Bolikowski, and M. Łukasik, “Evaluation of Features for Author Name Disambiguation Using Linear Support Vector Machines,” 2012.

[102] A. Al-Anazi and I. D. Gates, “A support vector machine algorithm to classify lithofacies and model permeability in heterogeneous reservoirs,” *Eng. Geol.*, vol. 114, no. 3–4, pp. 267–277, 2010.

[103] J. Kim and J. Kim, “The impact of imbalanced training data on machine learning for author name disambiguation,” *Scientometrics*, vol. 117, no. 1, pp. 511–526, 2018.

[104] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, “The influence of preprocessing on text classification using a bag-of-words representation,” *PLoS One*, vol. 15, no. 5, pp. 1–22, 2020.

[105] Li, Yin, Shi, Mao, and Shi, “Method of Feature Reduction in Short Text Classification Based on Feature Clustering,” *Appl. Sci.*, vol. 9, no. 8, p. 1578, Apr. 2019.

[106] G. A. Lujan-Moreno, P. R. Howard, O. G. Rojas, and D. C. Montgomery, “Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study,” *Expert Syst. Appl.*, vol. 109, pp. 195–205, Nov. 2018.

[107] H. Han, H. Zha, and C. L. Giles, “Name disambiguation spectral in author citations using a k-way clustering method,” in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries, JCDL, Denver, CO, USA*, pp. 7–11.

[108] J. Kim and J. Kim, “The impact of imbalanced training data on machine learning for author name disambiguation,” *Scientometrics*, vol. 117, no. 1, pp. 511–526, 2018.

PUBLIKASI

1. Firdaus, Firdaus, et al. "Improving Data Integrity of Individual-based Bibliographic Repository Using Clustering Techniques." Computer Engineering and Applications Journal 7.1 (2018): 45-50.

2. Firdaus, Firdaus, et al. "Deep Neural Network Structure to Improve Individual Performance based Author Classification." Computer Engineering and Applications Journal 8.1 (2019): 77-83.

3. Firdaus, Firdaus, et al. “Author Identification in Bibliographic Data Using Deep Neural Networks.”, TELKOMNIKA Telecommunication, Computing, Electronics and Control 19.3(2021): 911-919.

4. Firdaus, Firdaus, et al. “Neural Network Technique with Deep Structure for Improving Author Homonym and Synonym Classification.” TELKOMNIKA Telecommunication, Computing, Electronics and Control 19.4(2021): 1208-1217.

5. Firdaus, Firdaus, et al. “Author Matching Classification on A Highly Imbalanced Bibliographic Data Using Cost-Sensitive Deep Neural Network.”, 3rd International Conference on Informatics, Multimedia, Cyber, and Information System 2021”

6. Firdaus, Firdaus, et al. " Author Classification on Bibliographic Data Using Capsule Networks Architeture", 2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2022),