JSI : Jurnal Sistem Informasi *(E-Journal), VOL.14, NO.2, Oktober 2022*
ISSN Print : 2085-1588
ISSN Online :2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index
email: jsifasilkom@unsri.ac.id

Clustering Data Bibliografi menggunakan Algoritma DBSCAN dengan *Author Matching Classifier* Berbasis *Deep Neural Network*

Ricy Firnando[1], Siti Nurmaini[2], Sukemi[3], Firdaus[4], Muhammad Naufal Rachmatullah[5]
[1,2,3] Program Studi Magister Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya
[4] Program Studi Sistem Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya
[5] Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Sriwijaya
[1]ricy.firnando@gmail.com
[2]sitinurmaini@gmail.com
[3]sukemiku66@gmail.com
[4]virdauz@gmail.com
[5]naufalrachmatullah@gmail.com

Abstrak

Ambiguitas nama penulis atau *author name ambiguity* sering kali menjadi masalah yang dapat mempengaruhi kualitas layanan *database* bibliografi. Untuk mengatasi masalah ambiguitas nama penulis maka diciptakanlah disambiguasi nama penulis atau *author name disambiguation*. Metode yang digunakan dalam *author name disambiguation* umumnya menangani masalah ambiguitas nama penulis dengan pendekatan *author matching, classification, dan clustering*. Beberapa penelitian menggabungkan beberapa pendekatan seperti menggunakan *author matching classifier* berbasis algoritma *random forest* untuk *pairwise classification* dan algoritma DBSCAN sebagai algoritma *clustering* namun masih belum mendapatkan hasil atau performa yang optimal. Pada penelitian ini dibangun sebuah model *author matching classifier* berbasis *deep neural network* yang kemudian diimplementasikan dalam algoritma *clustering* DBSCAN. Berdasarkan percobaan yang dilakukan menggunakan *dataset The Giles*, model *author matching cassifier* berbasis *deep neural network* yang kami usulkan dapat menghasilkan performa sebesar 95.99% untuk *pairwise classification* dan 97.23% untuk *clustering*.

Kata kunci: Disambiguasi nama penulis, Klastering, DBSCAN, *Author matching classifier*, *Deep Neural Network*

Abstract

Author name ambiguity is often became a problem that can affect the quality of bibliographic data*base services. To solve the problem of author name ambiguity, author name disambiguation was created. The method used in author name disambiguation generally handles the problem of author name ambiguity with author matching, classification, and clustering approaches. Several studies have combined several approaches such as using an author matching classifier based on the random forest algorithm for pairwise classification and the DBSCAN algorithm as a clustering algorithm but have not yet obtained optimal results or performance. In this study, an author matching classifier model based on a deep neural network was built which was then implemented in the DBSCAN clustering algorithm. Based on experiments conducted using The Giles dataset, our proposed model can achieve a performance of 95.99% for pairwise classification and 97.23% for clustering.*

Keywords*: Author Name Disambiguation, Clustering, DBSCAN, Author matching classifier, Deep Neural Network*

1. PENDAHULUAN

Bibliographic Data*base* (BD) adalah basis data penyimpanan digital terorganisir yang berisi data bibliografi, paten, buku, artikel berita, dan lain-lain. Contoh BD yang umum digunakan adalah DBLP, CiteSeer, MEDLINE, Google Scholar, dan lain-lain. BD mencakup sejumlah besar data dari jaringan *author* dan perpustakaan digital. Dalam BD,

satu *author* dapat memiliki nama yang mirip dengan *author* yang lain, dan setiap *author* dapat memiliki berbagai cara untuk menulis nama lengkap mereka. Situasi ini menimbulkan ambiguitas bagi sistem yang membutuhkan data bibliografi sehingga mempengaruhi kualitas layanan sistem tersebut. Oleh karena itu pengidentifikasian secara unik untuk membedakan karya antara satu *author* dengan *author* lain sangatlah diperlukan, dan proses ini dikenal sebagai *Author Name Disambiguation* (AND). [1]

AND adalah proses membedakan *author* antara yang satu dengan yang lain. Tugas AND adalah menemukan kutipan mana yang dimiliki oleh *author* yang sama di antara banyaknya kutipan dari banyak *author*. Beberapa teknik telah diusulkan untuk AND dan beberapa penelitian di bidang AND menggunakan algoritma *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) untuk mengelompokkan data bibliografi berdasarkan identitas *author*.

Algoritma DBSCAN yang diterbitkan pada konferensi data *mining* KDD'96 adalah algoritma *clustering* berbasis kepadatan yang populer dan telah berhasil digunakan di banyak aplikasi dunia nyata. [2] Ada banyak metode berbasis kepadatan yang terinspirasi atau berdasarkan ide-ide yang diterbitkan di algoritma DBSCAN dan mencoba untuk memperbaiki keterbatasannya. [3]–[8] Algoritma DBSCAN terbukti berfungsi dalam praktik, dan pada tahun 2014, algoritma ini menerima penghargaan uji waktu SIGKDD.

Salah satu penelitian di bidang AND yang mengimplementasikan algoritma DBSCAN berhasil meraih nilai *f1* sebesar 95% untuk *pairwise classification* menggunakan *random forest classifier* dan 79% untuk *clustering*. [7] Namun nilai tersebut masih kurang optimal karena masih menimbulkan banyak kesalahan dalam pengidentifikasian dan pengelompokan nama *author* sehingga diperlukan peningkatan kinerja untuk mendapatkan hasil yang lebih optimal. Salah satu cara yang bisa digunakan untuk meningkatkan kinerja *pairwise classification* dan *clustering* pada DBSCAN adalah dengan menggunakan *author matching classifier* berbasis *Deep Neural Network* (DNN) seperti yang dilakukan pada penelitian AND menggunakan DNN yang berhasil meraih nilai akurasi sebesar 99,31% untuk *pairwise classification*. [9]

Pada penelitian ini akan dibangun sebuah algoritma DBSCAN yang mengimplementasikan *author matching classifier* berbasis DNN untuk mengelompokkan data bibliografi ke dalam sejumlah *cluster* berdasarkan identitas *author*. Alasan penulis memilih *author matching classifier* berbasis DNN untuk diimplementasikan dalam algoritma DBSCAN adalah karena berdasarkan referensi pada penelitian ini, *author matching classifier* tersebut menghasilkan nilai akurasi yang cukup tinggi dan berpotensi meningkatkan kinerja *clustering* saat diimplementasikan dalam algoritma DBSCAN sehingga dapat meminimalisir kesalahan pengidentifikasian dan pengelompokan nama *author*.

2. METODOLOGI PENELITIAN

Tahap-tahap yang dilakukan dalam penelitian ini adalah menyiapkan *dataset*, melakukan *preprocessing* terhadap data di dalam *dataset*, mencari jarak / kemiripan antara semua pasangan data di dalam *dataset*, membagi data jarak / kemiripan menjadi data *training* dan data *testing*, membuat model *author matching classifier* menggunakan data *training*, mengukur kinerja model *author matching classifier* menggunakan data *testing*, mengimplementasikan model *author matching classifier* ke dalam algoritma

DBSCAN, melakukan *clustering* data, dan mengukur kinerja *clustering* data. Tahapan dalam penelitian ini ditunjukkan pada gambar 1.

Gambar 1. Tahapan Penelitian

2.1. Persiapan *Dataset*

Dataset yang digunakan dalam penelitian ini adalah *Dataset The Giles* yang didapat dari perpustakaan digital DBLP dan sudah melalui *cleaning process* untuk membuang data dari *author* yang jumah data bibliografinya kurang dari 5 sehingga data yang awalnya berjumlah 5018 kemudian berkurang menjadi 4419. *Dataset* ini memiliki 6 atribut yaitu *Author Name*, *Author List*, *Title*, *Venue*, *Year*, dan *Author_id*.

2.2. *Preprocessing*

Preprocessing dilakukan terhadap *dataset* bertujuan untuk menghasilkan data yang relevan dan terstruktur sehingga diharapkan mampu menghasilkan sistem dengan komputasi yang efisien. *Preprocessing* yang dilakukan dalam penelitian ini antara lain:

1. *Case folding*: mengubah semua huruf pada *text* menjadi bentuk huruf yang sama. Dalam penelitian ini *case folding* yang akan digunakan adalah mengubah semua huruf pada *text* menjadi huruf kecil (*lower case*)
2. *Punctuation removal*: menghapus tanda baca pada *text*. Tanda baca yang dihapus adalah karakter selain huruf ataupun angka seperti titik (.) , koma (,) , tanda tanya (?) tanda seru (!) , dan lain-lain.
3. *Stop word removal*: menghapus kata-kata umum yang biasanya tidak memiliki informasi penting dan diabaikan dalam pemrosesan *text*.

4. *Stemming* adalah proses menghapus imbuhan *suffix* atau *prefix* dari suatu kata sehingga kata tersebut 2871tatist ke bentuk dasarnya.

2.3. Kombinasi Data

Kombinasi data adalah proses mengkombinasikan data-data di dalam *dataset*. Data pertama akan dikombinasikan dengan data kedua, kemudian data pertama dikombinasikan dengan data ketiga dan seterusnya. Sebagai contoh, jika ada 4 data yaitu A, B, C, dan D, maka akan ada 6 kombinasi data yaitu AB, AC, AD, BC, BD, dan CD. Jumlah kombinasi data dapat dihitung dengan persamaan (1).

$$C = \frac{(n*n)-n}{2} \quad \text{.....(1)}$$

Keterangan:
C : Jumlah kombinasi data
n : Jumlah data

Dataset the giles yang digunakan dalam penelitian ini memiliki 4419 data sehingga jika dilakukan kombinasi menggunakan persamaan (1) maka akan menghasilkan 9.761.571 kombinasi data.

2.4. Hitung Jarak atau Kemiripan

Jarak atau kemiripan setiap kombinasi data dihitung dengan cara membandingkan semua atribut dari data yang dikombinasikan. Sebagai contoh perbandingan data bibliografi A dan B serta cara perbandingan yang digunakan adalah sebagai berikut:

1. Jarak atribut *author name* : *jaccard* (*author name* (A), *author name* (B))
2. Jarak atribut *author list* : *jaccard* (*author list* (A), *author list* (B))
3. Jarak atribut *title* : *jaccard* (*title* (A), *title* (B))
4. Jarak atribut *venue* : *jaccard* (*venue* (A), *venue* (B))
5. Jarak atribut *year* : *absolute* (*year* (A), *year* (B))
6. Jarak atribut *author id* : *equal* (*author id* (A), *author id* (B))

Jaccard adalah 2871tatistic yang digunakan untuk mengukur kemiripan (*similarity*) dan jarak (*distance*) antar himpunan data. *Jaccard similarity coefficient* dinotasikan dengan persamaan (2) dan *Jaccard distance* dinotasikan dengan persamaan (3).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad \text{.....(2)}$$

$$d_j(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \quad \text{.....(3)}$$

Absolute adalah nilai mutlak dari hasil pengurangan dua buah bilangan atau selisih antara dua buah bilangan. Dalam penelitian ini *absolute* digunakan untuk menghitung jarak atribut *year*. *Equal* dalam penelitian ini adalah perbandingan sederhana. Operasi ini akan menghasilkan nilai 1 jika *author id* (A) sama dengan *author id* (B) dan akan menghasilkan nilai 0 jika *author id* (A) berbeda dengan *author id* (B).

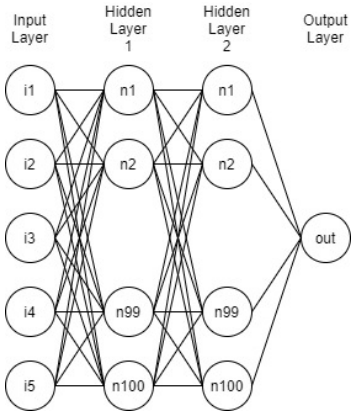
2.5. Pembagian Data

Pembagian data yang dilakukan dalam penelitian ini bertujuan untuk menentukan data *training* yang digunakan untuk membangun *author matching classifier* dan data *testing* yang akan digunakan untuk validasi *author matching classifier*. Pada penelitian ini dipilih 80% kombinasi data secara acak sebagai data *training* dan sisanya yaitu 20% kombinasi data dipilih sebagai data *testing*. Dengan jumlah kombinasi data sebanyak 9.761.571 kombinasi data maka pembagian data pada penelitian ini akan menghasilkan 7.809.257 data *training* dan 1.952.314 data *testing*.

Data-data yang akan dijadikan sebagai fitur dalam membangun *author matching classifier* adalah jarak atribut *author name*, jarak atribut *author list*, jarak atribut *title*, jarak atribut *venue*, dan jarak atribut *year*. Sedangkan data yang akan dijadikan sebagai label adalah jarak atribut *author id.*

2.6. Bangun *Author Matching Classifier*

Author matching classifier adalah model yang melakukan prediksi terhadap dua data bibliografi. Model ini menerima *input* berupa data jarak kemiripan dari dua data bibliografi dan menghasilkan *output* berupa prediksi yang menyatakan apakah dua data bibliografi tersebut merupakan milik *author* yang sama atau berbeda. Pada tahap ini, data *training* digunakan untuk membangun model *author matching classifier* menggunakan algoritma DNN dengan 1 *layer input* yang memiliki 5 *node*, 2 *hidden layer* yang masing masing memiliki 100 *node* dan menggunakan fungsi aktivasi *ReLu*, serta 1 *output layer* yang memiliki 1 *node* dan menggunakan fungsi aktivasi *sigmoid*. Arsitektur *deep neural network* yang digunakan pada penelitian ini dapat dilihat pada gambar 2.



Gambar 2. Arsitektur DNN

Hasil dari tahap ini adalah model *author matching classifier* yang menerima *input* berupa jarak semua atribut dari dua data bibliografi dan menghasilkan *output* berupa nilai *probability* dan *class*. *Probability* adalah nilai kemiripan antara dua data input dalam bentuk nilai 0-1. *Class* adalah nilai *Boolean* yang merepresentasikan apakah dua data yang diinputkan adalah milik author yang sama. Nilai *class* didapatkan dengan menerapkan *threshold default* terhadap nilai *probability*, jika nilai *probability* lebih besar atau sama dengan *threshold* maka nilai *class* akan bernilai 1 sedangkan jika nilai *probability* lebih kecil dari *threshold* maka nilai *class* akan bernilai 0.

2.7. Validasi *Author Matching Classifier*

JSI : Jurnal Sistem Informasi *(E-Journal), VOL.14, NO.2, Oktober 2022*
ISSN Print : 2085-1588
ISSN Online :2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index
email: jsifasilkom@unsri.ac.id

Validasi model *author matching classifier* berbasis DNN dilakukan dengan cara memproses data *testing* dan membandingkan output dari *author matching classifier* berupa nilai *class* dengan label yang berupa jarak atribut *author id*. Proses validasi ini akan menghasilkan 4 kondisi yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) seperti yang ditunjukkan oleh tabel 1.

Tabel 1. Kondisi *Output Author matching classifier*

Kondisi	*Class*	*Label*
True Positive (TP)	1	1
True Negative (TN)	0	0
False Positive (FP)	1	0
False Negative (FN)	0	1

Output author matching classifier kemudian disajikan dalam bentuk *confusion matrix*, yaitu tabel yang digunakan untuk mengukur kinerja *author matching classifier*. *confusion matrix* yang digunakan dalam penelitian ini ditunjukkan pada tabel 2.

Tabel 2. *Confusion Matrix Output Author matching classifier*

		Actual Values (Label)	
		***Negative* (0)**	***Positive* (1)**
Predicted Values (Class)	***Negative* (0)**	*True Negative* (TN)	*False Negative* (FN)
	***Positive* (1)**	*False Positive* (FP)	*True Positive* (TP)

Nilai-nilai pada *confusion matrix* kemudian digunakan untuk menghitung *performance metric* berupa *accuracy*, *precision*, *recall*, dan *f1* menggunakan persamaan (4) – (7).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \ldots\ldots\ldots(4)$$

$$Precision = \frac{TP}{TP+FP} \quad \ldots\ldots\ldots(5)$$

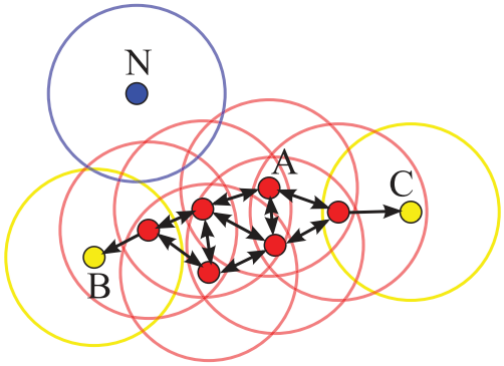
$$Recall = \frac{TP}{TP+FN} \quad \ldots\ldots\ldots(6)$$

$$F1 = \frac{2*precision*recall}{precision+recall} \quad \ldots\ldots\ldots(7)$$

2.8. Implementasi *Author Matching Classifier* pada DBSCAN

Algoritma DBSCAN adalah salah satu algoritma *clustering* berbasis kepadatan. Algoritma ini dapat mendeteksi jumlah *cluster* secara otomatis. DBSCAN mengidentifikasi setiap *cluster* sebagai area dengan kepadatan data yang tinggi (*dense region*) yang dipisahkan oleh area dengan kepadatan data yang rendah. DBSCAN menentukan *dense region* berdasarkan parameter *epsilon* dan *minimum point*. *Epsilon*

adalah jarak neighborhood suatu data dan *minimum point* adalah jumlah data minimal yang harus ada di dalam *neighborhood* suatu data sebagai syarat bagi data tersebut untuk menjadi *core point*. *Core point* adalah data yang di dalam *neighborhood*-nya terdapat data minimal sebanyak *minimum point*. Selain *core point* ada juga *border point* dan *noise*. *Border point* adalah data yang bukan merupakan *core point* tapi berada di dalam *neighborhood core point* sedangkan *noise* adalah data yang bukan merupakan *core point* ataupun *border point*. Model *cluster* DBSCAN dapat dilihat pada gambar 3.



Gambar 3. Model Cluster Algoritma DBSCAN [10]

Gambar 3 mencontohkan konsep dari algoritma DBSCAN dengan parameter *Minimum Points* bernilai 4, radius *epsilon* diilustrasikan dengan lingkaran, A adalah *core points*, B dan C adalah *border points*, dan N adalah *noise*. Algoritma DBSCAN akan mengidentifikasi semua *core point* yang ada di dalam data, kemudian menggabungkan setiap *core point* yang saling terhubung menjadi *cluster*. Untuk setiap titik yang bukan merupakan *core point*, jika titik tersebut terhubung dengan salah satu *core point* maka titik tersebut akan digabungkan ke dalam *cluster* sebagai *border point*, sedangkan jika titik tersebut tidak terhubung dengan *core point* maka titik tersebut akan diberi label sebagai *noise*.

Author matching classifier berbasis DNN dapat diimplementasikan dalam algoritma DBSCAN sebagai parameter *epsilon*. Jika output *author matching classifier* untuk data A dan B adalah 1 maka data A berada di dalam *neighborhood* data B dan data B juga berada di dalam *neighborhood* data A, sedangkan jika output *author matching classifier* untuk data A dan B adalah 0 maka data A berada di luar *neighborhood* data B dan data B juga berada di luar *neighborhood* data A.

3. HASIL DAN ANALISA

3.1. Hasil *Author Matching Classifier*

Model *author matching classifier* dilatih menggunakan algoritma *deep neural network* kemudian digunakan untuk melakukan *pairwise classification* terhadap data *testing*. *Pairwise classification* kemudian menghasilkan data berupa nilai *True Positive* (TP) sebanyak 15.207 data, *True Negative* (TN) sebanyak 1.935.839 data, *False Positive* (FP) sebanyak 502 data, dan *False Negative* (FN) sebanyak 767 data. Nilai-nilai tersebut

kemudian disajikan dalam bentuk *confusion matrix* seperti yang ditunjukkan pada tabel 3.

Tabel 3. *Confusion Matrix Pairwise Classification* menggunakan *Author matching classifier* berbasis *Deep Neural Network*

		Actual Values (Label)	
		***Negative* (0)**	***Positive* (1)**
Predicted Values (Class)	***Negative* (0)**	1.935.839	767
	***Positive* (1)**	502	15.207

Nilai-nilai pada *confusion matrix* kemudian digunakan untuk menghitung *performance metric* berupa *accuracy*, *precision*, *recall*, dan *f1* menggunakan persamaan (4) - (7) dan menghasilkan nilai seperti yang ditunjukkan pada tabel 4.

Tabel 4. *Performance Metric Pairwise Classification* menggunakan *Author matching classifier* berbasis *Deep Neural Network*

No.	***Metric***	***Score***
1	*Accuracy*	0.999350002
2	*Precision*	0.968043797
3	*Recall*	0.951984475
4	*F1*	0.959946975

3.3. Hasil *Clustering*

Clustering menggunakan DBSCAN dilakukan dengan mengimplementasikan model *author matching classifier* berbasis *deep neural network* sebagai fungsi jarak untuk mencari *neighbourhood* setiap data. Hasil *clustering* kemudian dianalisa untuk mengetahui berapa banyak data yang berhasil dikelompokkan ke dalam *cluster* yang tepat. Pada penelitian ini, model *author matching classifier* berbasis *deep neural network* digunakan sebagai fungsi jarak pada DBSCAN untuk mengelompokkan data pada *dataset The Giles*. *Clustering* menghasilkan 302 *cluster* dengan 97.23% data berhasil dikelompokkan pada *cluster* yang tepat.

4. KESIMPULAN

Berdasarkan hasil yang didapat, bisa dilihat bahwa nilai *F1* untuk *pairwise classification* yang dihasilkan oleh *author matching classifier* berbasis *deep neural network* berhasil mengungguli nilai *F1* untuk *pairwise classification* yang dihasilkan oleh

author matching classifier berbasis *random forest* yaitu 95.99 % berbanding 95%. Hal ini dikarenakan *deep neural network* dapat melakukan prediksi dengan performa yang lebih baik jika dibandingkan dengan *random forest*. Sedangkan untuk *clustering*, algoritma DBSCAN yang menggunakan *author matching classifier* berbasis *deep neural network* juga berhasil unggul dengan nilai 97.23% berbanding 79%.

DAFTAR PUSTAKA

[1] M. Shoaib, A. Daud, and T. Amjad, "Author Name Disambiguation in Bibliographic Databases: A Survey," Apr. 2020, Accessed: Oct. 10, 2020. [Online]. Available: http://arxiv.org/abs/2004.06391.

[2] M. Ester, M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," pp. 226--231, 1996, Accessed: Oct. 31, 2020. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220.

[3] "An efficient approach to *clustering* in large multimedia databases with noise | Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining." https://dl.acm.org/doi/10.5555/3000292.3000302 (accessed Nov. 05, 2020).

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS," in *Proceedings of the 1999 ACM SIGMOD international conference on Management of data - SIGMOD '99*, 1999, pp. 49–60, doi: 10.1145/304182.304187.

[5] E. Biçici and D. Yuret, "Locally scaled density based *clustering*," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, vol. 4431 LNCS, no. PART 1, pp. 739–748, doi: 10.1007/978-3-540-71618-1_82.

[6] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based *clustering* based on hierarchical density estimates," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7819 LNAI, no. PART 2, pp. 160–172, doi: 10.1007/978-3-642-37456-2_14.

[7] M. Khabsa, P. Treeratpituk, and C. L. Giles, "Online Person Name Disambiguation with Constraints," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, Jun. 2015, vol. 2015-June, pp. 37–46, doi: 10.1145/2756406.2756915.

[8] K. Kim, M. Khabsa, and C. L. Giles, "Inventor name disambiguation for a patent database using a random forest and DBSCAN," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2016, vol. 2016-Septe, pp. 269–270, doi: 10.1145/2910896.2925465.

[9] H. N. Tran, T. Huynh, and T. Do, "Author name disambiguation by using *deep neural network*," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8397 LNAI, no. PART 1, pp. 123–132, doi: 10.1007/978-3-319-05476-6_13.

[10] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Jul. 2017, doi: 10.1145/3068335.