Penentuan Matriks Similaritas Protein Menggunakan *Transfer Learning*

(*Determination of Protein Similarity Matrix using Transfer Learning*)

Yustina Sri Suharini*, Endang Ratnawati D, Sulistyowati, Muhamad Ramli

Program Studi Teknik Informatika, Institut Teknologi Indonesia,
Jl. Raya Puspitek, Serpong, Kota Tangerang Selatan, Banten 15320

Abstrak

Penjajaran sekuen protein merupakan hal yang penting untuk menentukan kesamaan antara protein satu dengan protein lainnya. Namun algoritma penjajaran sekuen yang sudah ada mempunyai tingkat kompleksitas tinggi sehingga eksekusi program pencarian kesamaan protein memerlukan waktu yang lama apabila jumlah sekuen yang disejajarkan sangat banyak. Penelitian ini bertujuan mendapatkan matriks similaritas protein dengan metode eksperimen berbasis transfer learning dengan arsitektur transformer. Data yang digunakan sebagai masukan untuk transformer adalah data sekuen protein berformat teks. Hasil penelitian berupa matriks similaritas protein yang dapat digunakan oleh para peneliti lain di bidangnya masing-masing sebagai bahan untuk dianalisis lebih lanjut.

Kata Kunci : similaritas protein, *transfer learning*, transformer

Abstract

Protein sequence alignment is important to determine the similarity between one protein and another. However, existing sequence alignment algorithms have a high level of complexity so that the execution of the protein similarity search program takes a long time if the number of sequences being aligned is very large. This research aims to obtain a protein similarity matrix using a transfer learning-based experimental method with transformer architecture. The data used as input for the transformer is protein sequence data in text format. The research results are in the form of a protein similarity matrix which can be used by other researchers in their respective fields as material for further analysis.

Keyword : *protein similarity*, *transfer learning*, *transformer*

* Penulis Korespondensi. Tepl:+62 21 7561092; fax: +62 21 7560542
Alamat E-mail : yustina.ss@iti.ac.id

1. Pendahuluan

Similaritas satu protein dengan protein lain merupakan hal penting, karena dapat digunakan untuk melakukan studi atau pendekatan terhadap kompleksitas protein yang belum sepenuhnya terungkap. Beberapa kegunaan utamanya antara lain untuk memahami fungsi protein, termasuk memprediksi fungsi, memahami struktur protein, mendesain obat dan terapi, mempelajari evolusi protein, membandingkan spesies dan kekerabatan, mencari gen, serta klasifikasi gen.

Salah satu cara yang dapat dilakukan untuk menentukan similaritas protein adalah dengan menjalankan algoritma penjajaran sekuen, atau yang dikenal dengan istilah *sequence alignment*. Namun algoritma-algoritma penjajaran sekuen yang sudah ada mempunyai tingkat kompleksitas tinggi yaitu O(nXm) dengan n dan m masing-masing adalah panjang sekuen query dan sekuen referensi. Atau dapat juga dituliskan secara umum bahwa tingkat kompleksitasnya $O(n^2)$. Algoritma yang dimaksud adalah Needleman-Wunsch untuk penjajaran global, dan algoritma Smith-Waterman

untuk penjajaran lokal. Banyak usaha dari peneliti untuk menurunkan tingkat kompleksitas algoritma penjajaran global dengan mengeksekusinya secara paralel menggunakan *graphical processing unit* (GPU) [1]–[7]. Banyak peneliti juga melakukan paralelisasi terhadap algoritma penjajaran lokal [8]–[19]. Namun persoalannya adalah, komputer untuk melakukan komputasi paralel berskala besar seperti itu tidak murah.

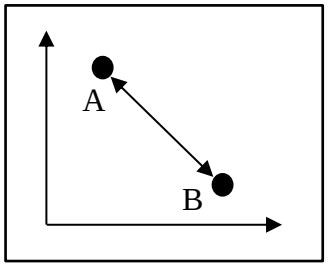
Di sisi lain kecerdasan buatan mengalami perkembangan pesat dalam tahun-tahun terakhir, termasuk pengenalan bahasa alami. Oleh karena itu penelitian ini bertujuan memanfaatkan model pengenalan bahasa alami untuk membuat matriks similaritas protein. Jika sekuen protein dianggap sebagai kalimat-kalimat yang diperlakukan seperti halnya bahasa alami, maka kompleksitas protein juga dapat didekati makna utuhnya menggunakan mekanisme pengenalan leksikal, konteks, serta semantik. Penelitian ini dilakukan menggunakan asumsi dasar tersebut.

2. Teori Dasar

Pada pengolahan bahasa alami, similaritas dua kalimat dapat didekati dengan fungsi jarak. Fungsi jarak yang paling umum dan dibayangkan dengan mudah adalah jarak *euclidian* (*euclidian distance*) dengan formula seperti ditunjukkan pada rumus (1). Jarak *euclidean* antara dua titik A dan B dalam ruang berdimensi k adalah panjang jalur yang membatasi titik pertama (A) dan titik kedua (B) dengan perhitungan jarak *pythagoras* [20].

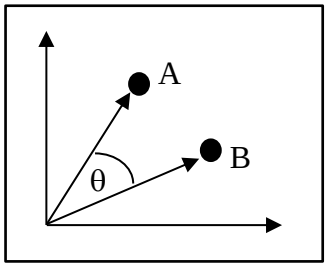
$$JE = \sqrt{\sum_{i=1}^{k}(X_i - Y_i)^2} \quad ....(1)$$

Seperti terlihat di rumus (1) bahwa jarak ini hanya mempertimbangkan selisih jalur antara satu titik dengan titik lain tanpa menggunakan titik pusat atau orientasi. Masalah akan muncul ketika terjadi penambahan jumlah titik-titik yang dibandingkan.



Gambar 1. Jarak *Euclidian* titik A dan B.

Sementara itu jarak *cosine* (*cosine distance*) mempunyai titik pusat atau titik orientasi untuk membandingkan dua buah titik yang berada di ruang berdimensi k. Pada formula jarak *cosine*, yang dijadikan elemen perhitungan adalah *cosine* sudut antara dua vektor yang dicari similaritasnya, yaitu *cosinus* (θ). Memperhatikan dua ilustrasi yang ditunjukkan pada Gambar 1 dan Gambar 2, maka jarak *cosine* dipilih karena mengakomodasi skalabilitas, dalam hal ini adalah jumlah vektor-vektor terkait protein, dibandingkan dengan jarak *euclidian*.



Gambar 2. Jarak *cosine* A dan B.

$$JC = \cos(\theta) = \frac{\vec{a}.\vec{b}}{||\vec{a}||\,||\vec{b}||} \quad ....(2)$$

Rumus (2) merupakan formula jarak *cosine*. Pembilang berupa perkalian sekalar antara vektor $\vec{a}$ dan vektor $\vec{b}$. Jika vektor $\vec{a} = ax\hat{\imath} + ay\hat{\jmath}$ dan vektor $\vec{b} = bx\hat{\imath} + by\hat{\jmath}$, maka perkalian sekalar vektor a dan vektor b adalah seperti rumus (3).

$$\vec{a}.\vec{b} = ax\ bx + ay\ by. \quad ....(3)$$

Penyebut pada rumus (2) adalah perkalian panjang vektor $\vec{a}$ dan vektor $\vec{b}$. Penentuan panjang vektor menggunakan rumus (4).

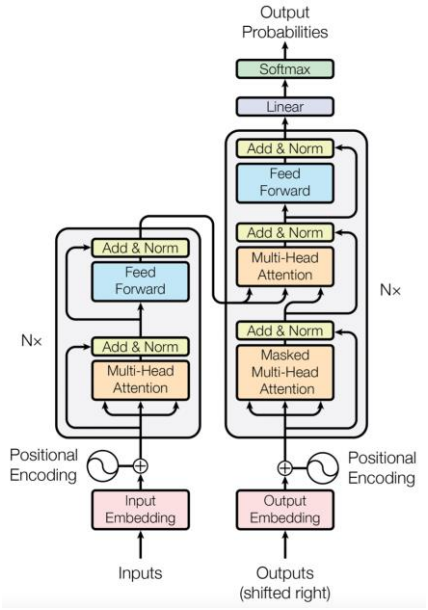
$$||\vec{a}|| = \sqrt{ax^2 + ay^2} \quad ....(4)$$

Apabila terdapat banyak dimensi, maka rumus (2) akan diakumulasi menjadi rumus (5).

$$\cos(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\left\|\sqrt{\sum_{i=1}^{n} {A_i}^2}\right\| \left\|\sqrt{\sum_{i=1}^{n} {B_i}^2}\right\|} \quad ....(5)$$

3. Metodologi

Penelitian bertujuan menggunakan langkah-langkah pemrosesan bahasa alami yang dalam tahun terakhir mengalami kemajuan sangat pesat, khususnya arsitektur model bahasa yang dikenal dengan nama BERT, singkatan dari *bidirectional encoder representation transformer* [21]. Kata pertama mempunyai arti bahwa pelatihan atau training dilakukan dua arah, dari kiri ke kanan dan dari kanan ke kiri, atau *bidirectional*. Kata kedua berarti *encoder* menerima masukan kalimat lalu melakukan *encode* masukan tersebut. Bentuk kata dalam kalimat direpresentasi sebagai vektor, maka kata ketiga adalah *representation*. Sedangkan kata keempat adalah *transformer*, yang artinya model menggunakan arsitektur *transformer*, sebuah arsitektur dengan komponen perhatian (*attention*) sebagai karakteristik pembeda dari model-model sebelumnya. Arsitektur *transformer* ditunjukkan pada Gambar 3 [22].



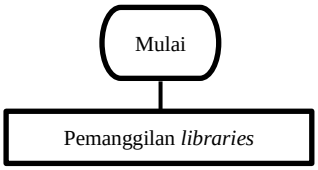
Gambar 3. Arsitektur transformer dari artikel "*Attention is All You Need*" [22].

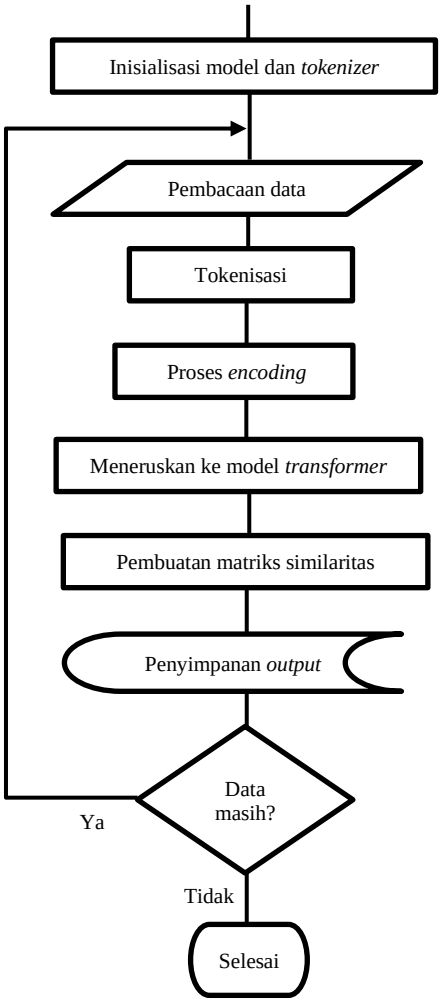
Langkah-langkah penelitian adalah sebagai berikut.

1. Pengumpulan data sekuan protein yang dikumpulkan dari protein databank (PDB), dipilih 20 sekuen dengan panjang yang berbeda-beda.
2. Pemilihan model *machine learning* berdasar hasil studi literatur.
3. Pembuatan kode program dengan melibatkan library *machine learning* yang sudah dipilih.
4. Uji coba kode program dengan sekuen pendek untuk memastikan *correctness* program.
5. Pengamatan hasil uji coba program.
6. Pembetulan program jika masih salah.
7. Kembali ke langkah 5, 6, 7 sampai program betul.
8. Eksekusi program untuk 20 data sekuen protein yang sesungguhnya.
9. Penyimpanan hasil eksekusi program.
10. Pembahasan dan dokumentasi.

Jalannya kode program yang dibuat dalam penelitian, dapat dilihat pada Gambar 4. Pertama, pemanggilan pustaka (*libraries*) yang diperlukan. *Library* utama adalah model transformer. Namun ada juga beberapa *library* lain yang harus diinstal untuk memperudah pelaksanaan eksekusi program serta pengamatan dan penyimpanan hasil.

Tabel 1 menunjukkan data sekuen protein yang digunakan untuk percobaan, beserta ukuran panjang masing-masing.





Gambar 4. *Flowchart* jalannya program.

Tabel 1. Daftar Sekuen

No.	ID Protein	Ukuran (Karakter)
1	UPI000DC7E9D1	3423
2	UPI000D4D4AD9	1274
3	UPI000DDB9683	1274
4	UPI000DC260E8	1282
5	UPI000D2DD5DD	1708
6	UPI000DC89C88	1285
7	UPI000DA773BC	2541
8	UPI000DFE3007	1402
9	UPI000DED18CD	1206
10	UPI000DA43176	1296
11	UPI000DA4D51F	1418
12	UPI000DC5385B	2575
13	UPI000D02674A	1517
14	UPI000D4D22E4	1837
15	UPI000D62AE66	2905
16	UPI000DE93595	1767
17	UPI000D8AA964	2007
18	UPI000D190F40	1705
19	UPI000D809524	1515
20	UPI000DBCB130	4217

4. Hasil dan Pembahasan

Score similaritas sekuen UPI000DC7E9D1 terhadap 20 sekuen yang sedang diselidiki dalam penelitian disajikan pada Tabel 2. Tabel tersebut memberi gambaran secara garis besar bahwa dari 20 sekuen yang digunakan, tidak ada satupun yang menghasilkan nilai similaritas negatif. Justru keduapuluh sekuen mempunyai tingkat kemiripan yang tinggi, berkisar dari 0,88 sampai 0,94 (tidak termasuk yang dibandingkan dengan diri sendiri, yang mana *score* similaritas 1).

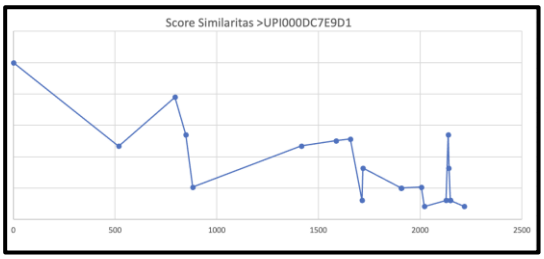
Tabel 2. *Score* Similaritas

No.	ID Protein	*Score* Similaritas terhadap UPI000DC7E9D1
1	UPI000DC7E9D1	1
2	UPI000D4D4AD9	0.89024
3	UPI000DDB9683	0.89024
4	UPI000DC260E8	0.91594017
5	UPI000D2DD5DD	0.89024
6	UPI000DC89C88	0.942636
7	UPI000DA773BC	0.9008429
8	UPI000DFE3007	0.88532794
9	UPI000DED18CD	0.88532794
10	UPI000DA43176	0.89024
11	UPI000DA4D51F	0.9008429
12	UPI000DC5385B	0.942636
13	UPI000D02674A	0.9000775
14	UPI000D4D22E4	0.9378119
15	UPI000D62AE66	0.9334953
16	UPI000DE93595	0.9391664
17	UPI000D8AA964	0.93361413
18	UPI000D190F40	0.91594017
19	UPI000D809524	0.9000775
20	UPI000DBCB130	0.9724185

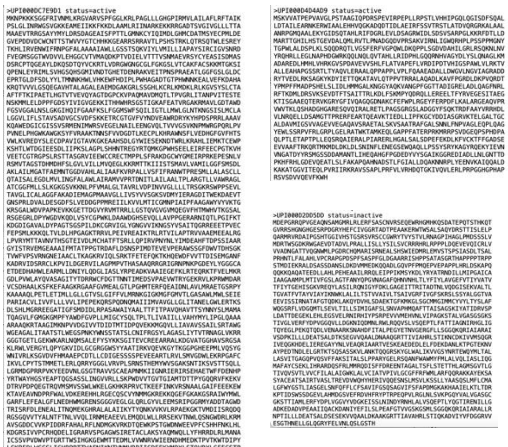
Tabel 3 adalah tabel yang berisi hubungan antara ukuran sekuen dengan *score* similaritasnya. Pada kasus yang panjang sekuennya sama, secara kebetulan score similaritasnya juga sama. Hal ini menimbulkan gagasan penelitian lanjutan dengan menggunakan sejumlah sekuen dengan panjang yang sama semuanya. Pada Gambar 5 ditunjukkan hubungan antara perbedaan ukuran sekuen dengan *score* similaritas pasangan sekuen.

Tabel 3. Ukuran dan Score Similaritas

No.	Ukuran	*Score* Similaritas terhadap Ukuran 3423
1	3423	1
2	1274	0.89024
3	1274	0.89024
4	1282	0.91594017
5	1708	0.89024
6	1285	0.942636
7	2541	0.9008429
8	1402	0.88532794
9	1206	0.88532794
10	1296	0.89024
11	1418	0.9008429
12	2575	0.942636
13	1517	0.9000775
14	1837	0.9378119
15	2905	0.9334953
16	1767	0.9391664
17	2007	0.93361413
18	1705	0.91594017
19	1515	0.9000775
20	4217	0.9724185



Gambar 5. Hubungan antara perbedaan ukuran sekuen dengan *score* similaritas.



Gambar 6. Contoh-contoh ukuran sekuen.

Waktu yang diperlukan untuk eksekusi program bervariasi tergantung panjang dua sekuen yang dimasukkan ke dalam program. Untuk 20 sekuen yang digunakan, waktu eksekusi sekitar beberapa puluh detik sampai mendekati 3 menit. Oleh karena itu ukuran sekuen menjadi penting diselidiki lebih lanjut. Untuk mempertegas hal itu, disajikan beberapa contoh sekuen dengan panjang yang beragam pada Gambar 6.

5. Kesimpulan

Beberapa hal pada percobaan yang menjadi simpulan penelitian adalah sebagai berikut.

1. Nilai similaritas yang didapat oleh program terhadap pasangan sekuen protein berkisar dari 0,88 sampai 0,94. Padahal ukuran panjang bervariasi. Itu artinya, similaritas tidak hanya ditentukan oleh panjang sekuen.

2. Tidak ditemukan hasil yang negatif pada data yang digunakan pada percobaan, berarti semua data mempunyai tingkat kemiripan yang relatif tinggi.
3. Waktu eksekusi program masih bergantung pada panjang sekuen. Jadi belum terjadi penurunan tingkat kompleksitas seperti yang diharapkan.
4. Perlu penelitian lanjutan untuk memastikan bahwa ada korelasi positif antara perbedaan panjang sekuen dengan *score* similaritas.

Ucapan Terima Kasih

Terima kasih kepada panitia Technopex Institut Teknologi Indonesia 2023 yang memberi kesempatan kepada kami untuk memaparkan hasil penelitian pada Seminar Nasional Technopex ITI tanggal 25 Oktober 2023.

Daftar Pustaka

[1] Y. S. Lee, Y. S. Kim, and R. L. Uy, "Serial and parallel implementation of Needleman-Wunsch algorithm," *International Journal of Advances in Intelligent Informatics*, vol. 6, no. 1, pp. 97–108, Mar. 2020, doi: 10.26555/ijain.v6i1.361.

[2] Da Li and M. Becchi, "Abstract: Multiple Pairwise Sequence Alignments with the Needleman-Wunsch Algorithm on GPU," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, IEEE, Nov. 2012, pp. 1471–1472. doi: 10.1109/SC.Companion.2012.267.

[3] C. Kyal, R. Kumar, and A. Zamal, "Performance-Based Analogising of Needleman Wunsch Algorithm to Align DNA Sequences Using GPU and FPGA," in *2020 IEEE 17th India Council International Conference, INDICON 2020*, IEEE, Dec. 2020, pp. 1–5. doi: 10.1109/INDICON49873.2020.9342078.

[4] A. Chaudhary, D. Kagathara, and V. Patel, "A GPU based implementation of Needleman-Wunsch algorithm using skewing transformation," in *2015 8th International Conference on Contemporary Computing, IC3 2015*, IEEE, Aug. 2015, pp. 498–502. doi: 10.1109/IC3.2015.7346733.

[5] H. Nadim, M. Assal, and A. A. Hegazy, "An efficient framework for accelerating Needleman-Wunsch algorithm using GPU," *Int J Bioinform Res Appl*, vol. 17, no. 2, pp. 101–110, 2021, doi: 10.1504/IJBRA.2021.114412.

[6] M. Fakirah, M. A. Shehab, Y. Jararweh, and M. Al-Ayyoub, "Accelerating Needleman-Wunsch global alignment algorithm with GPUs," in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, IEEE, Nov. 2015, pp. 1–5. doi: 10.1109/AICCSA.2015.7507113.

[7] M. Fakirah, M. A. Shehab, Y. Jararweh, and M. Al-Ayyoub, "Accelerating Needleman-Wunsch global alignment algorithm with GPUs," *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, vol. 2016-July. 2016. doi: 10.1109/AICCSA.2015.7507113.

[8] J. Pérez Serrano, E. F. De Oliveira Sandes, A. C. Magalhaes Alves de Melo, and M. Ujaldón, "Smith-Waterman acceleration in multi-GPUs: A performance per watt analysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10209 LNCS. pp. 512–523, 2017. doi: 10.1007/978-3-319-56154-7_46.

[9] A. Bustamam, G. Ardaneswari, H. Tasman, and D. Lestari, "Performance evaluation of fast smith-waterman algorithm for sequence database searches using CUDA GPU-based parallel computing," *Journal of Next Generation Information Technology*, vol. 5, no. 2, pp. 38–46, 2014, [Online]. Available: https://api.elsevier.com/content/abstract/scopus_id/84930012659

[10] X. Feng, H. Jin, R. Zheng, Z. Shao, and L. Zhu, "Implementing Smith-Waterman algorithm with two-dimensional cache on GPUs," *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*. pp. 25–30, 2012. doi: 10.1109/CGC.2012.98.

[11] K. Dohi, K. Benkrid, C. Ling, T. Hamada, and Y. Shibata, "Highly efficient mapping of the Smith-Waterman algorithm on CUDA-compatible GPUs," *Proceedings of the International*

Conference on Application-Specific Systems, Architectures and Processors. pp. 29–36, 2010. doi: 10.1109/ASAP.2010.5540796.

[12] C. Ling, K. Benkrid, and T. Hamada, "A parameterisable and scalable Smith-Waterman algorithm implementation on CUDA-compatible GPUs," in *2009 IEEE 7th Symposium on Application Specific Processors*, IEEE, Jul. 2009, pp. 94–100. doi: 10.1109/SASP.2009.5226343.

[13] M. J. Yin, X. Xu, Z. Xiong, T. Zhang, and F. Zheng, "An improved smith-waterman algorithm on heterogeneous CPU-GPU Systems," *International Journal of Applied Mathematics and Statistics*, vol. 50, no. 20, pp. 499–507, 2013, [Online]. Available: https://api.elsevier.com/content/abstract/scopus_id/84896782034

[14] E. F. D. O. Sandes and A. C. M. A. De Melo, "Smith-Waterman alignment of huge sequences with GPU in linear space," *Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011*. pp. 1199–1211, 2011. doi: 10.1109/IPDPS.2011.114.

[15] E. F. Edans and A. C. M. A. De Melo, "Retrieving smith-waterman alignments with optimizations for megabase biological sequences using GPU," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 1009–1021, 2013, doi: 10.1109/TPDS.2012.194.

[16] M. Yin, X. Xu, Z. Xiong, F. Zheng, and T. Zhang, "Optimizing Smith-Waterman algorithm based on CPU and GPU through CUDA platform," *International Review on Computers and Software*, vol. 7, no. 7, pp. 3627–3632, 2012, [Online]. Available: https://api.elsevier.com/content/abstract/scopus_id/84875160251

[17] D. V. V. Prasad and S. Jaganathan, "Improving the performance of Smith–Waterman sequence algorithm on GPU using shared memory for biological protein sequences," *Cluster Comput*, vol. 22, no. S4, pp. 9495–9504, Jul. 2019, doi: 10.1007/s10586-018-2421-7.

[18] Ł. Ligowski, W. R. Rudnicki, Y. Liu, and B. Schmidt, "Accurate scanning of sequence databases with the smith-waterman algorithm," *GPU Computing Gems Emerald Edition*. pp. 155–171, 2011. doi: 10.1016/B978-0-12-384988-5.00011-5.

[19] A. Ali and B. Ram, "GPU performance survey on OpenCL and CUDA using smith waterman algorithm," *International Journal of Applied Engineering Research*, vol. 10, no. 55, pp. 1320–1323, 2015, [Online]. Available: https://api.elsevier.com/content/abstract/scopus_id/84942465871

[20] B. Mumbai, *Mastering spaCy*, 2021.

[21] J. Devlin, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1. pp. 4171–4186, 2019. [Online]. Available: https://api.elsevier.com/content/abstract/scopus_id/85083815650

[22] A. Vaswani, "Attention Is All You Need," no. Nips, 2017.