

# Biomedical-named entity recognition using CUDA accelerated KNN algorithm

Manish Bali<sup>1</sup>, Anandaraj Shanthi Pichandi<sup>2</sup>, Jude Hemanth Duraisamy<sup>3</sup>

<sup>1</sup>Department of Computer Science and Engineering, Presidency University, Bengaluru-560064, Karnataka, India

<sup>2</sup>Department of Computer Science and Engineering, Presidency University, Bengaluru-560064, Karnataka, India

<sup>3</sup>Department of Electronics and Communication Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India

## Article Info

### Article history:

Received May 28, 2022

Revised Dec 29, 2022

Accepted Feb 16, 2023

### Keywords:

BioNLP

Graphics processing unit

Machine learning

Named entity recognition

Natural language processing

## ABSTRACT

Biomedical named entity recognition (Bio-NER) is a highly complex and time-consuming research domain using natural language processing (NLP). It's widely used in information retrieval, knowledge summarization, biomolecular event extraction, and discovery applications. This paper proposes a method for the recognition and classification of named entities in the biomedical domain using machine learning (ML) techniques. Support vector machine (SVM), decision trees (DT), K-nearest neighbor (KNN), and its kernel versions are used. However, recent advancements in programmable, massively parallel graphics processing units (GPU) hold promise in terms of increased computational capacity at a lower cost to address multi-dimensional data and time complexity. We implement a novel parallel version of KNN by porting the distance computation step on GPU using the compute unified device architecture (CUDA) and compare the performance of all the algorithms using the BioNLP/NLPBA 2004 corpus. Results demonstrate that CUDA-KNN takes full advantage of the GPU's computational capacity and multi-leveled memory architecture, resulting in a 35× performance enhancement over the central processing unit (CPU). In a comparative study with existing research, the proposed model provides an option for a faster NER system for higher dimensionality and larger datasets as it offers balanced performance in terms of accuracy and speed-up, thus providing critical design insights into developing a robust BioNLP system.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Manish Bali

Department of Computer Science and Engineering, Presidency University

Itgapur, Rajankunte, Yelahanka, Bengaluru-560064, Karnataka, India

Email: balimanish0@gmail.com

## 1. INTRODUCTION

Biomedical-named entity recognition (Bio-NER) is a sub-task of information extraction that classifies all unregistered terms found in biomedical texts. Location, person, organization, date, time, percentage, monetary value, and none of the above are the eight categories used by named entity recognition [1], [2]. It locates named entities in sentences and brackets them under one of the categories. For example, in the sentence: Apple (organization) CEO Tim Cook (person) introduces two new, larger iPhones, smartwatch at Cupertino (location), and Flint center (organization) event [3]. As captured against each of the words keeping the context of the sentence in view, instead of a fruit name apple has been recognized as an organization. A person's name is arrived at by combining Tim and Cook representing their chief executive officer (CEO) and so forth. Bio-NER assigns a named entity tag to a given word using rules and heuristics. To categorize a word into one of the following three categories: person, organization, or date, it is essential to identify the specified entity and to extract

symbolic and discernible information from a text [4], [5]. By using current categories and none of the above, named entity recognition categorizes every term in the document. This includes recognizing named entities such as proteins, deoxyribonucleic acid (DNA), ribonucleic acid (RNA), cells, and other key entities in biomedical literature. It is a hot topic of research, but the literature review shows a clear gap of at least 10% in accuracy when comparing the best-performing algorithms for Bio-NER with generic news or text [6]-[9].

Researchers studying natural language processing are particularly interested in the information extraction of genes, cancer, and proteins from biomedical literature. Although it is essential to extracting biological information, Bio-NER has only been treated as the first stage of text mining in biomedical literature [10]-[14]. Recognizing technical terms in the biomedical area has long been one of the most challenging problems in natural language processing related to biomedical research [15]. In contrast to the categories used in the conventional named entity recognition technique, in this paper, we use five categories to recognize a protein, DNA, RNA, cell line, and cell type. An example of a Bio-NER tagged sentence is: IL-2 (B-protein) responsiveness requires three distinct elements (B-DNA) within the enhancer (B-DNA). As seen from the above simple sentence, finding biomedical named entity recognition is difficult and complex from the text on account of the following reasons [16], [17]:

- Because of how quickly contemporary research is progressing, there are an increasing number of new technical terms. Thus, it is challenging to create a gazetteer that includes all the new jargon.
- The same words or expressions may be classified as different named entities depending on the context.
- An entity's length is quite long, and it may contain control characters like hyphens (for example, 12-O-Tetradecanoylphorbol-13-acetate).
- In the biomedical domain, abbreviated expressions are widely utilized, and they have a sense of ambiguity. Tissue culture fluid (TCF) might, for example, stand for T cell factor or tissue culture fluid (TCF) [18].
- In biomedical language, normal and functional meanings are combined or incorporated, which is why a phrase may become very long. Human T-lymphotropic virus I (HTLVI)-infected and HTLV-I-transformed, for example, comprise the phrases I, infected, and transformed.
- Biomedical NER has a hard time segmenting sentences with named entities. Changes in spelling can also be problematic.
- A named entity from one category may be subsumed by a named entity from another category.

As a result, experts continue to find this to be a complicated area. Three types of methodologies have been studied for identifying biological named entities: rule-based heuristics [19], [20], dictionary-based [21] approaches, and statistical machine learning techniques [22]. As many named entities as possible are stored in a gazetteer using a dictionary-based mechanism. Though straightforward, this method has its drawbacks. Firstly, since target terms are mostly proper nouns or unregistered words, named entity recognition is tough. Additionally, new words are produced often, and depending on the context, the same word stream may be recognized as a variety of named entities. A rule-based approach is based on the conventions and patterns of named entities that are present in sentences that are spoken. Although the problem of many named entities can be solved using context in rule-based procedures, each rule must be created before it is used. The third approach, based on machine learning, links the discovered items to words even though the words are not in the dictionary and the context is not provided in the rule set. In this domain, popular methods include support vector machine (SVM) [5], decision tree (DT) [23], Hidden Markov models (HMM) [18], [24], maximum entropy Markov model (MEMM) [25], and conditional random field (CRF) [24]. When training with test data that isn't part of the training data, supervised learning techniques use data with a variety of features based on several linguistic criteria. Mathu and Raimond [26] used Bi-LSTM with residuals for a drug-named entity recognition task with only a 40% recognition rate for 4-gram entities. Ramachandran and Arutchelvan [27] have implemented a hybrid model in which the annotated entities are trained by a blank Spacy ML model validated by this dictionary and human. The average F1 score was 73.79% [28] used the classical rule-based approach on PubMed articles achieved an F1 score of 77.65%. However, it has been noticed that the high-end systems built do not produce adequate results on multi-dimensional and large datasets because no single method can generate higher performance. There is a pressing need to address these issues. Graphic processing units (GPUs) on the other hand have become more programmable and emerged as a form of manycore processor with highly parallelized and multi-threaded capabilities during the last decade [29]. The modern GPU provides great computing horsepower and higher memory bandwidth when compared to generic x86-based central processing units (CPUs) at a fraction of the cost. The GPU is now at the cutting edge of chip-level parallelism, with applications ranging from 3D rendering [30] to general-purpose computing [31].

In this paper, three machine learning (ML) techniques from different paradigms are tested on the BioNLP/NLPBA 2004 corpus [32], [33], which contains 22,402 sentences. SVM, DT, and K-nearest neighbor (KNN) with two kernel versions i.e., KNN-1 and KNN-5 are implemented on a CPU. Among these, the KNN algorithm is a commonly used machine learning technique for pattern recognition, classification, and

regression [21], [34]. This algorithm is an instance based-lazy learning technique that is utilized in a variety of applications like image processing and pattern matching. Even though the algorithm is straightforward, it takes a while to run. The application's execution time may become a bottleneck when the train and test data sets are both very large. In this study, we describe a new parallel implementation of the KNN algorithm based on the compute unified device architecture (CUDA) and GPU from NVIDIA, and we compare the performance with that of conventional CPU execution. The research tries to investigate if this parallelized version of KNN surpasses other machine learning algorithms in terms of accuracy and speedup, which could provide interesting insights into the development of a reliable Bio-NER system.

Following are the sections of this manuscript: section 2 presents the methodology used in terms of explaining the three classifiers used and the dataset chosen, the basic principle of the KNN algorithm, and a note on NVIDIA GPU and CUDA technology and the programming approach adopted on the algorithm towards a KNN-CUDA version. Section 3 presents both the classification and execution time results and a comparison with other existing models along with a detailed analysis. Finally, the conclusion is covered in section 4 followed by references.

## 2. METHOD

Figure 1 shows a high-level flow chart of the complete procedure. The BioNLP/JNLPS 2004 corpus is the data set used. The data is pre-processed using natural language processing operations such as stop word removal, numeric value removal, tokenization, and lemmatization because it contains a lot of undesired data/noise that can affect the overall accuracy of the model. The preprocessed data is then transformed using the word2vec word embedding or feature extraction technique into numerical vectors. Before being fed to the machine learning models, the data is divided into train (2000 MEDLINE abstracts) and test (404 MEDLINE abstracts) data. Three popular machine learning algorithms, SVM, DT, and KNN are used for the Bio-NER classification of words in a sentence, and the efficacy of each technique is compared in the first part of the study. In the second part of the study, all the algorithms executed on a traditional CPU for efficacy are also checked on their execution time, which is captured in ms (milliseconds). The KNN distance computation is parallelized using NVIDIA CUDA GPU and a novel data segmentation approach as explained in section 2, section 3, and section 4.

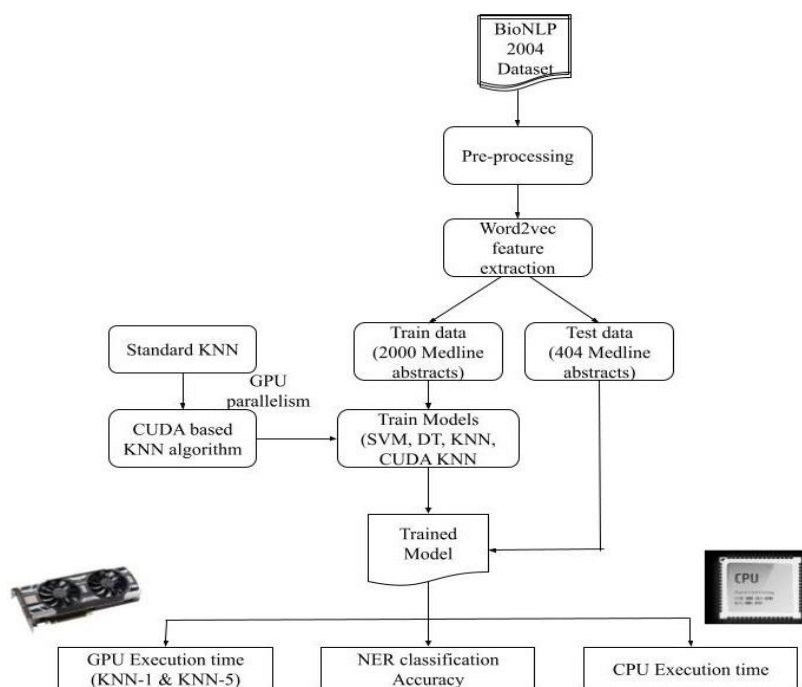


Figure 1. Flow-chart of the proposed methodology

This CUDA version of KNN is run on a CPU+GPU in the system and execution time is noted. This time is compared with the earlier CPU run and performance compared. The next sections provide a thorough explanation of the dataset used, the algorithm, and the parallelization of KNN.

### 2.1. Dataset

We use 22402 sentences in the BioNLP/NLPBA 2004 corpus [31], [32]. The training data set consists of 18546 and the test data consists of 3856 sentences. The dataset is further divided into groups like protein, DNA, RNA, cell line, and cell type.

### 2.2. Support vector machine

This approach for supervised machine learning is often used to solve classification and regression problems. It is a binary, linear, non-probabilistic classifier that makes use of data patterns. It reduces generalization error on unobserved examples by employing the structural risk minimization principle. As depicted in Figure 2, the objective of the SVM algorithm is to construct the optimal decision boundary, or hyperplane, that can divide  $n$ -dimensional space into classes and make it simple to classify fresh data points. It selects the extreme points or vectors called support vectors that aid in building the hyperplane. SVM uses the following function to predict:

$$Y' = w \times \phi(x) + b \quad (1)$$

The non-linear transform is stated in (1) by  $\phi(x)$ , with the primary goal of obtaining the appropriate values of  $w, b$ . In (1),  $Y'$  is obtained by minimizing the risk of regression.

$$R_{reg}(Y') = C \times \sum_{i=0}^l \gamma(Y'_i - Y_i) + \frac{1}{2} \times \|w\|^2 \quad (2)$$

Where:

$$w = \sum_{j=1}^l (\alpha_j - \alpha_j^*) \phi(x_j) \quad (3)$$

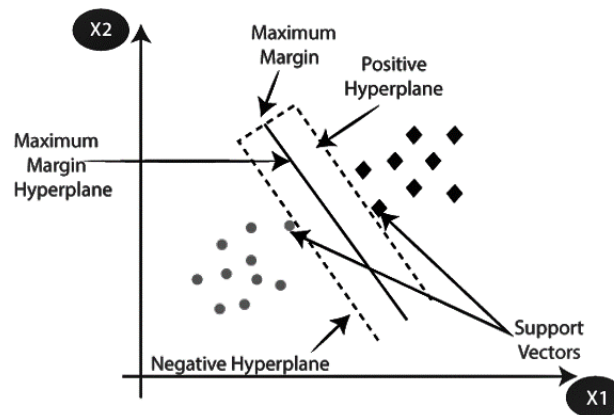


Figure 2. Illustration of support vector machine [35]

In (3), the parameters  $\alpha$  and  $\alpha^*$  state the relaxation parameter called the Lagrange multiplier. The output obtained is:

$$Y' = \sum_{j=1}^l (\alpha_j - \alpha_j^*) \phi(x_j) \times \phi(x) + b \quad (4)$$

$$Y' = \sum_{j=1}^l (\alpha_j - \alpha_j^*) \times K(x_j, x) + b \quad (5)$$

In (4) and (5),  $K(x_j, x)$  states the kernel function. To extract named entities, SVM with linear kernel function is used in this study.

### 2.3. Decision tree

A non-parametric supervised learning approach called a decision tree is employed for both classification and regression. Figure 3 depicts its hierarchical tree structure, which includes a root node, branches, internal nodes, and leaf nodes. We employed the C5.0 model, which performs recursive partitioning across the biological data, to extract event expressions. Starting at the root node, each node of the tree divides the feature vector or data into multiple branches based on the association rule between the split criteria. It is an inductive method that aids to acquire knowledge in classification.

## 2.4. K-nearest neighbor (KNN)

K-nearest neighbor is utilized for both classification and regression issues, and there is no training step; instead, the entire data set is used to predict/classify fresh data. As shown in Figure 4, when a new data point needs to be categorized, it determines how far apart from all other data points in categories A and B it is. Next it determines where the dataset's nearest neighbor(s) are based on the  $k$  value; if  $k = 1$ , all points are assigned to the same class as the data point with the smallest distance based on the lowest distance between them. A list of the  $k$  minimum distances between all of the data points is returned if  $k$  is higher than one.

### 2.4.1. Principle of KNN algorithm

According to the method, a query point must fall into a certain category if the majority of the  $k$  samples that are most similar to it in the feature space also do. This method is known as the KKN algorithm because the distance in the feature space can be used to assess similarity. A trained data set with precise classification labels should be known before the technique is started. Then determine the distance between each point in the train data set and a query data point  $qi$ , whose label is unknown and which is represented by a vector in the feature space. The labels of the  $k$  closest points in the train data set may be used to decide the test point  $qi$ 's class label after sorting the results of the distance computation.

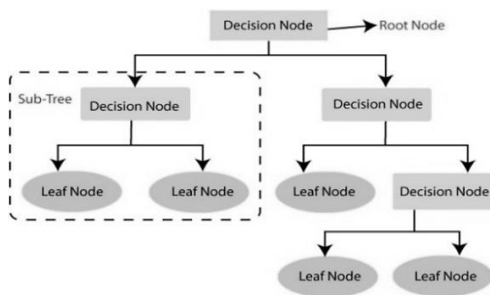


Figure 3. Illustration of a decision tree [36]

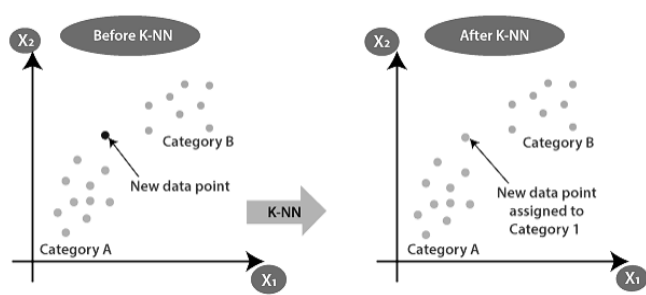


Figure 4. Illustration of KNN algorithm [37]

Each point in  $d$ -dimensional space can be represented as a  $d$ -vector of coordinates, as represented in (6).

$$p = (p_1, p_2, \dots, p_n) \quad (6)$$

The distance between two points in a multidimensional feature space can be specified in a variety of ways. The Euclidean distance is frequently calculated [37] using (7).

$$\text{dist}(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (7)$$

The quality of the training dataset has a direct impact on the classification outcomes. The choice of the parameter  $k$  is crucial since various  $k$  values can lead to various classification labels. In this study,  $k = 1$  and  $k = 5$  are used. High-dimensional data sets can be handled via the KNN approach, which is simple to use. The computational complexity is significant and the operation time is considerable when the test set, training set, and data dimension are higher than projected. When the test and training sets consist of  $m$  and  $n$  vectors in  $d$ -dimensional feature space, respectively, the temporal complexity of this method is  $O(m \times n \times d)$ .

### 2.4.2. NVIDIA GPU and CUDA

Theoretically, GPU performance outperforms CPU performance by a wide margin. This is because GPUs are built for computationally demanding, highly parallel computations, like those involved in rendering graphics. Because of the enormous processing capacity provided by GPU at a relatively low cost, researchers have experimented to use a GPU for non-graphics tasks. This effort was called a general-purpose graphics processing unit (GPGPU). Yet, it was challenging to design GPUs for non-graphics applications before the introduction of CUDA. For general-purpose computation on its GPUs, NVIDIA's CUDA is a parallel programming language. With the help of CUDA, programmers can accelerate computationally extensive parts of the algorithm by making use of GPUs' capacity to parallelize this portion of the code. Three key abstractions form the basis of the CUDA parallel programming concept:

- A hierarchy of thread groups
- Shared memories
- Synchronization of barriers

The CUDA abstractions, which are nested beneath coarse-grained data parallelism and task parallelism, enable fine-grained data parallelism and thread parallelism. They demonstrate to the programmer how to break the problem down into coarse sub-problems that blocks of threads can solve independently in parallel and finer sub-problems that the threads in the block can solve jointly in parallel. Figure 5(a) demonstrates how a kernel is executed concurrently by a collection of threads. Each thread executes the same code and has an ID that it uses to calculate memory addresses and make control decisions. Threads are arranged into a grid of thread blocks, as shown in Figure 5(b). Different kernels may employ various block configurations. The shared memory that is accessible to threads from the same block allows for synchronized execution. Thread blocks are necessary for each thread to run separately. These must be carried out in any arrangement, whether sequential or parallel. Programmers can organize thread blocks in any order across any number of cores, allowing them to create code that scales with the number of cores. Threads can cooperate within a block by exchanging information via shared memory and synchronizing their execution to manage memory accesses. Grids of blocks and thread blocks, as shown in Figure 6, might be one, two, or three-dimensional.

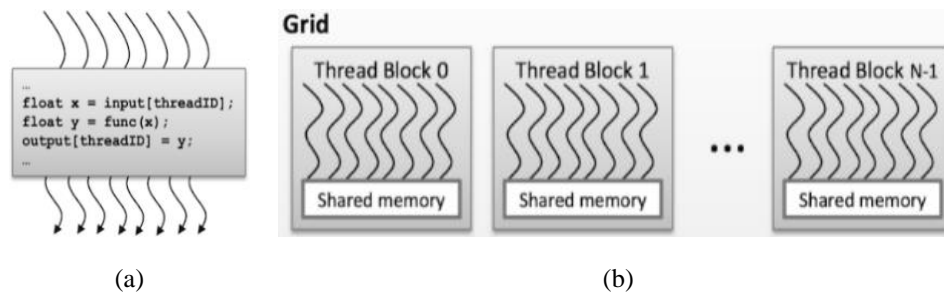


Figure 5. Depiction of (a) CUDA kernel and (b) a grid of CUDA thread blocks

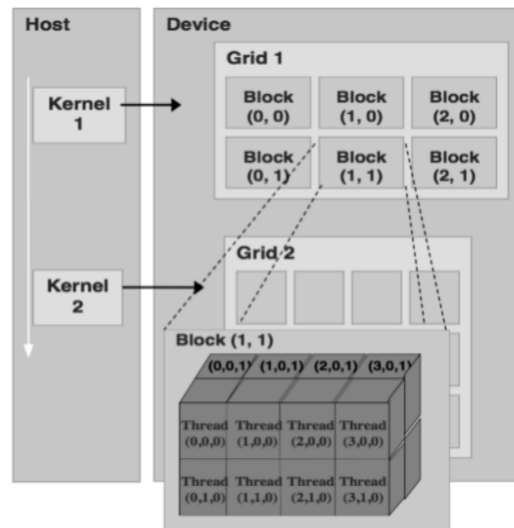


Figure 6. Thread organization in CUDA

### 2.4.3. KNN algorithm in CUDA

The KNN algorithm can be parallelized in CUDA using the fundamental steps listed below. The initialization of the test data set and the labeled  $d$ -dimensional train data set is the first stage in the pre-processing of the data. Next, determine the distances between one test point and each train data point using a sample from the test data set.

The results of distance computation are sorted (using CUDA-based radix sort) in the following step, and the  $k$  smallest results are chosen based on the parameter  $k$ . The test point's class label is determined in the fourth step by using the election result of  $k$  points (simple statistical election). A different point in the test data set should then be chosen, and step two repeated until the test data set is empty. The Euclidean distance is applied in this study.

A data segmentation approach is used to calculate distance. Each vector in the data set needs  $d$  features to represent it because both the train data set  $A$  and the test data set  $B$  are  $d$ -dimensional. We restore the train data set as an  $n \times d$  matrix in memory and the test data set as an  $m \times d$  matrix if the train data set has  $n$  vectors and the test data set has  $m$  vectors. The result set  $C$ , which includes all the distances between each pair of points in  $A$  and  $B$ , is described by a  $mn$  matrix. The distance between the vector in  $A$  with row number  $x$  and the vector in  $B$  with row number  $y$  is therefore represented by a column  $x$  and row  $y$  element in data set  $C$ . The distances recovered in  $C$  are squared Euclidean distances because the computation of the square root has no impact on the outcomes of the sorting.

There are many  $T$ -width tiles in the final data set  $C$ . The GPU's threads each handle one  $C$  element and compute distance, with the results from all threads being combined in the output. The CUDA kernel function pseudo algorithm is described in Table 1.

Table 1. Algorithm for distance computation (train, test, result)
{In 2D, each thread in each block is designated by the letters $bx$ , $by$ , $tx$ , and $ty$ }
$sub\_result = 0;$
$temp = 0;$
for each subtitle in dimension/ $T$ do
loads $shared\_train$ in the train set to shared memory;
loads $shared\_test$ in the test set to a shared memory;
sync threads;
for $K = 0$ to $T$ do
$temp = shared\_test[ty][k] shared\_prob[tx][k];$
$sub\_result += temp * temp;$
end for
sync threads;
add $sub\_results$ to the corresponding position in the result set;
end for

### 3. RESULTS AND DISCUSSION

The experimental setup used for the study is based on an Intel i7, 4-core, 8-thread processor-based system. The CPU-only runs use this configuration. The CUDA version of the KNN algorithm looks out for a GPU and the system has a NVIDIA GeForce MX450 2GB GDDR5 graphics memory card with 896 CUDA cores. The CPU off-loads the most compute-intensive part of the algorithm like the distance calculation onto the GPU which uses the inherent parallelism to accelerate this computation and provide the interim result back to the CPU which compiles the overall results.

#### 3.1. Classifier performance on CPU

Five classes of biomedical named entities, protein, DNA, RNA, cell line, and cell type were tagged in the first part of the study using the BioNLP/NLPBA 2004 dataset. The GENIA version 3.02 corpus, which contains 2,000 MEDLINE abstracts, is where the train data came from. 404 MEDLINE abstracts from the GENIA study were used for testing. The findings are presented as F1-scores, which are determined by applying (8):

$$F1 - score = (2 \times PR) / (P + R) \quad (8)$$

Where  $R$  stands for recall and  $P$  stands for precision.  $P$  is the quantity of correctly found named entity chunks divided by the quantity of correctly found named entity chunks, and  $R$  is the quantity of correctly found named entity chunks divided by the quantity of truly found named entity chunks. When every component of the recognized named entity has been correctly identified, it is regarded as the correct named entity. Figure 7(a) displays the results of the four different individual classifiers on the test corpus. The best F1-score is 71.17 percent for the decision tree model, followed by 70.16 percent for the KNN-5 model. SVM has a score of 68.02 percent, whereas KNN-1 has a score of 67.11 percent.

Considering the timings, Figure 7(b) illustrates the time taken by these classifiers on the test corpus. The support vector machine method, followed by decision tree and K-nearest neighbor algorithms, requires the least amount of CPU time. The distance computation phase of the K-nearest neighbor (KNN-5) model consumes the greatest CPU time. We select different feature sets for these classifiers since the sensitivity and effectiveness of each classifier vary. KNN models employ local features, full-text features, and a fraction of external resource features; the DT model mostly uses local features, whereas SVM models only use token features due to SVM's inefficiency. There is no post-processing applied to these models. The performance of the CUDA KNN method on this dataset is examined in the following section.

### 3.2. CUDA accelerated KNN performance

GPU and CPU refer to the CUDA implementation and the serial CPU approach compared in this paper respectively. The initialization and output components of the program are essentially identical. The core components of KNN algorithms' execution times are shown in Figure 8(a) and in comparison to other algorithms in Figure 8(b). As can be seen from the numbers, the GPU technique has a distinct advantage in terms of overall execution time. An execution time speedup of 32× is achieved on the CUDA implementation of the KNN-1 algorithm as compared to the CPU algorithm execution time, while an execution time speedup of 35× is reached on the CUDA implementation of KNN-5 algorithm when compared to the CPU approach execution time. This is because, in CPU implementation, the distance computing step takes up most of the execution time, which gets accelerated on GPU implementations the most.

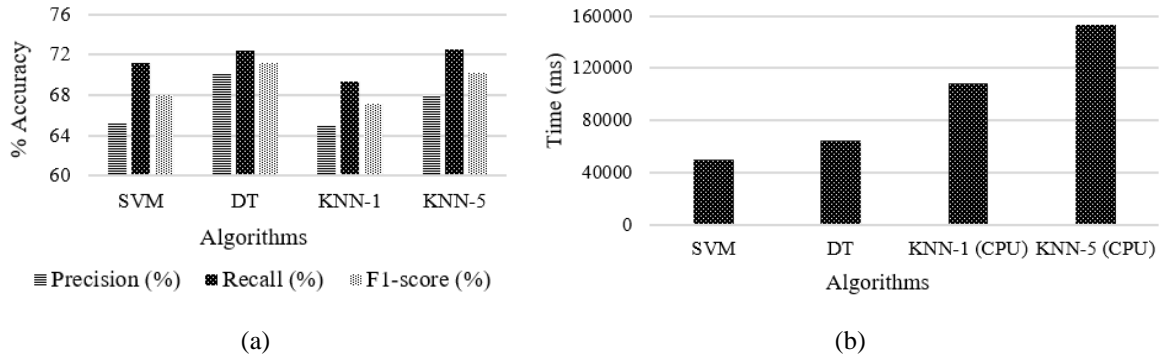


Figure 7. Classifier performance results on CPU and test corpus in terms of (a) accuracy and (b) execution time in ms

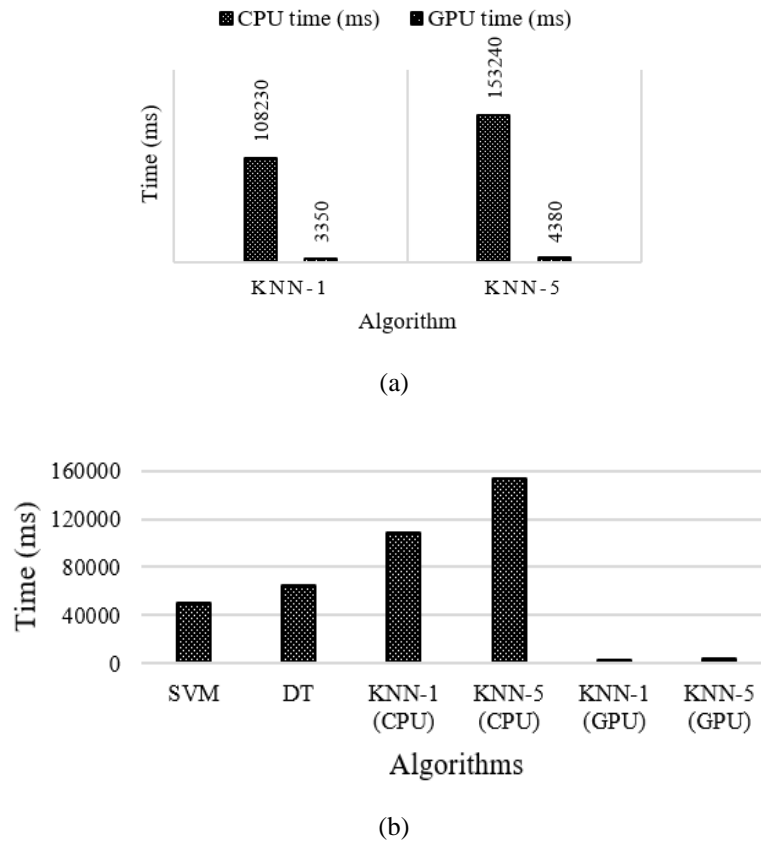


Figure 8. Result on GPU: (a) KNN algorithm execution time and (b) execution time for all algorithms in ms



As a classification algorithm, the approaches' efficiency should be mandatorily considered, as well as if GPU implementation has had any effect on F1-score. However, since (i) their principles are identical and (ii) they are trained for the same number of epochs, the classification accuracy of both approaches is always similar. The only difference and advantage are that the higher-performance GPU version of KNN finishes faster, allowing for a larger data set to be processed in the same amount of time. If we train all the algorithms for the same amount of time, the accuracy will vary because the higher-performance GPU version of KNN will complete a larger number of training iterations.

### 3.3. Performance comparison with existing models

Table 2 compares the results achieved with state-of-the-art existing models on the common BioNLP/NLPBA 2004 corpus to check their efficacy with the proposed models. The F1-score of the DT algorithm is 71.17%, which is the highest of all the findings. This is because it compels the analysis of every possible decision consequence and tracks every path to a conclusion. It generates a thorough analysis of the outcomes along each branch and indicates the decision nodes that need more research. KNN-5 comes in second with 70.16%, followed by SVM with 68.02%. Maximum entropy was utilized by [25] to acquire an F1-score of 67.41%. Using SVM and CRF [32] were able to attain a score of 66.28%. HMM was used by [28], who achieved a 65.7 percent score. The models proposed by Krauthammer and Nenadic [19] are also compared, with F1 scores of 60.75 and 58.80 percent, respectively. Although the CUDA version of KNN performs second best, its faster execution time and accuracy provide significant insights into developing a robust and fast-named entity recognition system for larger data sets. Also, it holds steady for more recent hybrid [27] and classical rule-based [28] approaches.

Table 2. Performance comparison with existing models

Reference research	Methodology	F1-score (%)	Time taken (in ms)
[19]	Jordan type	60.75	NA
[19]	Elman type RNN	58.80	NA
[32]	SVM, CRF	66.28	NA
[28]	HMM	65.70	NA
[25]	Maximum entropy	67.41	NA
Proposed models	Decision tree	71.17	65050
	SVM	68.02	49860
	KNN-1, GPU	67.11	3350
	KNN-5, GPU	70.16	4380

## 4. CONCLUSION

Since technical terms in biological texts have distinctive properties, Bio-NER has more challenges than regular NER. In the first study, we compared the performance of three machine-learning approaches from various paradigms and measured their execution time on a standard CPU. We next GPU or CUDA enables the distance computation and present the GPU-version of the KNN algorithm in the second study. Results show that the CUDA enabled solution fully utilizes the GPU's computational capabilities and multi-leveled memory architecture, resulting in a 35× speedup, a notable performance improvement over the traditional CPU implementation. And this is achieved on a sub-hundred-dollar graphics card, which is worth noting. The suggested GPU version of the KNN algorithm is valuable for applications having high dimensionality and large amounts of data. Comparing proposed models with existing research, though DT returns the best results in terms of accuracy in the true sense, the GPU version of KNN with 35× speedup over a CPU and near-best levels of accuracy is a good option to consider in building a robust and fast NER system for large multi-dimensional datasets. As part of a future study, we intend to investigate deep neural networks for biomolecular event extraction tasks for which named entity recognition is a sub-task. To develop a quick and efficient BioNLP system, paradigms like transfer learning and bidirectional encoder representations from transformers (BERT) models will also be investigated along with other newer hardware.

## REFERENCES




- [1] P. Bose, S. Srinivasan, W. C. Sleeman, J. Palta, R. Kapoor, and P. Ghosh, "A Survey on Recent Named Entity Recognition and Relationship Extraction Techniques on Clinical Texts," *Applied Sciences*, vol. 11, no. 18, 2021, doi: 10.3390/app11188319.
- [2] B. Alshaikhdeeb and K. Ahmad, "Biomedical Named Entity Recognition: A Review," *International Journal on Advanced Science Engineering and Information Technology*, vol. 6, no. 6, pp. 889-895, 2016, doi: 10.18517/ijaseit.6.6.1367.
- [3] K. B. Cohen and L. Hunter, "Natural language processing and systems biology," *Artificial Intelligence Methods and Tools for Systems Biology, Computational Biology*, Dordrecht: Springer, 2004, vol. 5, pp. 145-173, doi: 10.1007/978-1-4020-5811-0\_9.
- [4] E. F. T. K. Sang and F. D. Meulder, "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition," *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*, 2003, pp. 142-147, doi: 10.48550/arXiv.cs/0306050.

- [5] T. Meenachisundaram and M. Dhanabalachandran, "Biomedical Named Entity Recognition Using the SVM Methodologies and bio Tagging Schemes," *Revista de Chimie*, vol. 72, no. 4, pp. 52-64, 2021, doi: 10.37358/RC.21.4.8456.
- [6] A. Ekbal, S. Saha, and U. K. Sikdar, "Biomedical named entity extraction: some issues of corpus compatibilities," *SpringerPlus*, vol. 2, no. 601, 2013, doi: 10.1186/2193-1801-2-601.
- [7] X. Wang, C. Yang, and R. Guan, "A comparative study for biomedical named entity recognition," *International Journal of Machine Learning and Cybernetics*, vol. 9, pp. 373-382, 2018, doi: 10.1007/s13042-015-0426-6.
- [8] M. C. Cariello, A. Lenci, and R. Mitkov, "A Comparison between Named Entity Recognition Models in the Biomedical Domain," *Translation and Interpreting Technology Online*, pp. 76-84, 2021, doi: 10.26615/978-954-452-071-7\_009.
- [9] T. M. Thiyagu, D. Manjula, and S. Shridhar, "Named Entity Recognition in Biomedical Domain: A Survey," *International Journal of Computer Applications*, vol. 181, no. 41, 2019, doi: 10.5120/ijca2019918469.
- [10] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun, "A practical part-of-speech tagger," in *Proceedings of the third conference on applied natural language processing*, 1992, pp. 133-140, doi: 10.3115/974499.974523.
- [11] A. R. Aronson, T. C. Rindflesch, and A. C. Browne, "Exploiting a large thesaurus for information retrieval," *RIA0 '94: Intelligent Multimedia Information Retrieval Systems and Management*, vol. 94, pp. 197-216, 1994. [Online]. Available: <https://hncbc.nlm.nih.gov/ii/information/Papers/riao94.final.pdf>
- [12] P. G. Baker, C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass, "An ontology for bioinformatics applications," *Bioinformatics*, vol. 15, no. 6, pp. 510-530, 1999, doi: 10.1093/bioinformatics/15.6.510.
- [13] N. Perera, M. Dehmer, and F. E. -Streib, "Named Entity Recognition and Relation Detection for Biomedical Information Extraction," *Frontiers in Cell and Developmental Biology*, vol. 8, no. 673, 2020, doi: 10.3389/fcell.2020.00673.
- [14] K. Wang *et al.*, "NERO: a biomedical named-entity (recognition) ontology with a large, annotated corpus reveals meaningful associations through text embedding," *npj systems biology and applications*, vol. 7, no. 38, 2021, doi: 10.1038/s41540-021-00200-x.
- [15] C. Blaschke, M. A. Andrade, C. Ouzounis, and A. Valencia, "Automatic extraction of biological information from scientific text: protein-protein interactions", *Proceedings International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 60-67. [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/10786287/>
- [16] H. Liu, Z. Zhi, M. Torii, C. Wu, and C. Friedman, "Quantitative assessment of dictionary-based protein named entity tagging," *Journal of the American Medical Informatics Association*, vol. 13, no. 5, pp. 497-507, 2006, doi: 10.1197/jamia.M2085.
- [17] N. Ponomareva, F. Pla, A. Molina, and P. Rosso, "Biomedical named entity recognition: a poor knowledge HMM-based approach," *International Conference on Application of Natural Language to Information Systems*, 2007, vol. 4592, pp 382-387, doi: 10.1007/978-3-540-73351-5\_34.
- [18] M. Craven and J. Kumlien, "Constructing biological knowledge bases by extracting information from text sources," in *Proceedings International Conference on Intelligent Systems for Molecular Biology*, 1999, vol. 7, pp. 77-86. [Online]. Available: <https://www.biostat.wisc.edu/~craven/papers/ismb99.pdf>
- [19] M. Krauthammer and G. Nenadic, "Term identification in the biomedical literature," *Journal of Biomedical Informatics*, vol. 37, no. 6, pp. 512-538, 2004, doi: 10.1016/j.jbi.2004.08.004.
- [20] P. D. Soomro, S. Kumar, Banbharni, A. A. Shaikh, and H. Raj, "Bio-NER: Biomedical Named Entity Recognition using Rule-Based and Statistical Learners," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 12, 2017, doi: 10.14569/IJACSA.2017.081220.
- [21] K. Fukuda, A. Tamura, T. Tsunoda, and T. Takagi, "Toward information extraction: identifying protein names from biological papers," *Pacific Symposium Biocomputing*, 1998, pp. 707-725. [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/9697224/>
- [22] D. Campos, S. Matos, and J. L. Oliveira, "Biomedical Named Entity Recognition: A Survey of Machine-Learning Tools," *Theory and Applications for Advanced Text Mining*, United Kingdom: IntechOpen, 2012, doi: 10.5772/51066.
- [23] A. S. Al-Hegami, A. M. F. Othman, and F. Bagash, "A Biomedical Named Entity Recognition Using Machine Learning Classifiers and Rich Feature Set," *International Journal of Computer Science and Network Security*, vol. 17, no. 1, 2017. [Online]. Available: <https://www.semanticscholar.org/paper/A-Biomedical-Named-Entity-Recognition-Using-Machine-Al-Hegami-Othman/e6e2d078d06fcc25f51a866b4e45f1400a4c56e2>
- [24] S. Zhao, "Named entity recognition in biomedical texts using an HMM model," in *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, 2004, pp. 87-90. [Online]. Available: <https://aclanthology.org/W04-1216>
- [25] S. K. Saha, S. Sarkar, and P. Mitra, "Feature selection techniques for maximum entropy based biomedical named entity recognition," *Journal of Biomedical Informatics*, vol. 42, no. 5, pp. 905-916, 2009, doi: 10.1016/j.jbi.2008.12.012.
- [26] T. Mathu and K. Raimond, "A novel deep learning architecture for drug named entity recognition", *Telecommunication, Computing, Electronics and Control (TELKOMNIKA)*, vol. 19, No. 6, pp. 1884-189, 2021, doi: 10.12928/telkomnika.v19i6.21667.
- [27] R. Ramachandran and K. Arutchelvan, "Named entity recognition on bio-medical literature documents using hybrid based approach," *Journal of Ambient Intelligence and Humanized Computing*, 2021, doi: 10.1007/s12652-021-03078-z.
- [28] R. Ramachandran and K. Arutchelvan, "ArRaNER: A novel named entity recognition model for biomedical literature documents," *The Journal of Supercomputing*, vol. 78, pp. 16498-16511, 2022, doi: 10.1007/s11227-022-04527-y.
- [29] R. Kobus, T. Hundt, A. Müller, and B. Schmidt, "Accelerating metagenomic read classification on CUDA-enabled GPUs," *BMC Bioinformatics*, vol. 18, no. 11, 2017, doi: 10.1186/s12859-016-1434-6.
- [30] C. Xie, S. L. Song, J. Wang, W. Zhang, and X. Fu, "Processing-in-Memory Enabled Graphics Processors for 3D Rendering," *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 637-648, doi: 10.1109/HPCA.2017.37.
- [31] D. Zhao and Q. Chen, "Current Prediction Model of GPU Oriented to General Purpose Computing," in *IEEE Access*, vol. 7, pp. 127920-127931, 2019, doi: 10.1109/ACCESS.2019.2939256.
- [32] Y. Song, E. Kim, G. G. Lee and B. -K. Yi, "POSBOTM-NER in the shared task of BioNLP/NLPBA," in *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, 2004, pp. 100-103. [Online]. Available: <https://aclanthology.org/W04-1220.pdf>
- [33] K. Hakala and S. Pyysalo, "Biomedical Named Entity Recognition with Multilingual BERT," in *Proceedings of the 5th Workshop on BioNLP Open Shared Tasks*, 2019, pp. 56-61, doi: 10.18653/v1/D19-5709
- [34] S. Wang *et al.*, "kNN-NER: Named Entity Recognition with Nearest Neighbour Search," *Computation and Language*, 2022, doi: 10.48550/arXiv.2203.17103.
- [35] A. Yenikar, C. N. Babu, and D. J. Hemanth, "Figure 5: Illustration of support vector machine," *PeerJ Computer Science*, 2020, doi: 10.7717/peerj-cs.1100/fig-5.




- [36] A. Yenikar, C. N. Babu, and D. J. Hemanth, "Figure 4: Decision tree used for sentiment classification," *PeerJ Computer Science*, 2020, doi: 10.7717/peerj-cs.1100/fig-4.
- [37] JavaTpoint, "K-Nearest Neighbour (KNN) Algorithm for Machine Learning-Javatpoint." [www.javatpoint.com](http://www.javatpoint.com). <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning> (accessed Nov. 15, 2022).

## BIOGRAPHIES OF AUTHORS






**Manish Bali**    received a B.E. degree in electronics and communications engineering from Bangalore University, in 1995, and a master's degree in computer science from Ramaiah University of Advanced Sciences, in 2021. He has 28+ years of Industry experience and has worked as an Adjunct Professor at Presidency University, Bengaluru. He is currently a CSE research scholar, pursuing Ph.D. He has published 7 papers in various national and International Journals and has a patent to his name His current research interests include machine learning, deep learning, and emerging technologies. He can be contacted at email: [balimanish0@gmail.com](mailto:balimanish0@gmail.com).



**Anandaraj Shanthi Pichandi**    received the received B. E (CSE) degree from Madras University, Chennai in the year 2004, M.Tech (CSE) with Gold Medal from Dr. MGR Educational and Research Institute, University in the year 2007 (Distinction with Honor), and Ph.D. in August 2014. He is presently working as Professor under CSE at Presidency University, Bengaluru. He has 15+ Years of Teaching Experience. His areas of interest include Information security, Data Science, Machine Learning, and Networks. He has written two book chapters in IGI Global Publications, USA, and authored five books. He has filed 7 patents and published 50+ papers. He is a member of ISTE, CSI, Member of IACSIT, and Member of IAENG. He can be contacted at email: [anandsofttech@gmail.com](mailto:anandsofttech@gmail.com).



**Jude Hemanth Duraisamy**    received a B.E. degree in ECE from Bharathiar University, in 2002, an M.E. degree in communication systems from Anna University, in 2006, and the Ph.D. degree from Karunya University, Coimbatore, India, in 2013. He is currently working as an Associate Professor with the Department of ECE, at Karunya University. He has authored more than 130 research articles in reputed SCIE-indexed international journals and Scopus-indexed international conferences. His cumulative impact factor is more than 160. He has published 33 edited books with reputed publishers, such as Elsevier, Springer, and IET. His research interests include computational intelligence and image processing. He can be contacted at email: [judehemanth@karunya.edu](mailto:judehemanth@karunya.edu).