Similarity Detection among student's queries: A Text Mining approach

Kavita S. Oza, P. G. Naik R.K.Kamat

Abstract

Due to the pandemic situation it is difficult for student to query about admissions at university or colleges, as they cannot visit the college premises physically. Proposed research work focuses on processing student online queries related to admission process using text mining techniques. A query dataset along with the response to the query is developed i.e. query corpus. When students request a query, it is matched with query dataset for similarity. Here five different similarity measures viz. Jaccard, Cosine, Euclidean, Minkowski and Manhattan similarity measures are used. As per the similarity measures response is generated for students query. If query did not match with any record in the dataset then it is added to the dataset with response manually. It is observed that Cosine similarity gives more accurate similarity as compared to other similarity measures which gave approximate similarity among queries.

Keywords—Similarity measure, Text mining, Minkowski similarity, Jaccard similarity, Cosine similarity, Euclidean distance

1. Introduction

Covid pandemic has made almost everyone techno savvy. Due to restrictions on physical movement of people lots of new challenges have arose. One of them is the admission process of University and Colleges. As soon as the graduation results are declared students rush to different colleges or Universities to find out about programmes offered and admission process followed by them. Though all this data is available on the websites of respective universities and Institutes, students have some queries which are not listed and needs human interaction to answer. Many colleges have their chat boats hosted on their websites to provide online help. But they are not customized to answer local level language query. For example while asking for admission process of MCA course the query may be

" What is the process for admission?", "Admission process", "Admission?", "MCA", "Admission process keva?", "Admission process kaise hoga?" etc. Here a query needs to be mapped with approximate similarity measures.

Natural language processing is the most interesting area of research due to ambiguity in the data. Student's queries are in natural language which needs to be processed to make it suitable for applying text mining techniques. Text mining techniques are popularly used in document classification, document summarization, short answer analysis etc. One of the most used techniques of text mining is similarity measures between two sentences, which involve investigation of similarity between words of the sentences. Words can be lexically similar if they have all characters appearing in the same sequence. String based algorithms can be used for lexical analysis. In case of semantic similarity, it is knowledge based similarity. Here semantic networks are used to derive the similarity measures. This paper addresses both lexical as well as semantic similarity to map student query's to query dataset.

Rest of the paper is organized as: second section focuses on literature review; third section discusses implementation of proposed system, and four talks about conclusion and future extension.

2. Literature Review

Text mining plays an important role in natural language processing. It helps in interpreting the meaning, context , similarity etc. of the sentence or paragraph or a whole article. A number of researchers have carried out in-depth survey of text mining techniques (Kaushik A, et al. 2016; Ramya R. S, et al. 2017). Text mining techniques are also used to search about particular learning approach presence, in set of research articles (Salloum S. A, et al. 2018). Text mining tools are also successfully used in retrieving experiences from similar cases in effective designing of green buildings. (Shen L, et al. 2017). Automation of text analysis can help in speedy analysis of huge number of textual reports for geological knowledge extraction (Holden E. J, et al. 2019). Text mining also has great application in Science & technology and innovation in extending the quantitative methods applied there (Ranaei S, et al. 2019).

Text similarity is subpart of text mining which deals with two types of similarities. First one is lexical similarity, where two pieces of text are matched without considering their meaning; it is just word to word matching. Second semantic similarity considers the meaning of the words while calculating similarity matrix. Semantic technology can also be used in identifying academic content similarity (Saquicela V, et al. 2018). Semantic similarities technique is also to develop patent lane (Niemann, et al. 2017). Semantic similarity among contextual words and informative words in knowledge graphs using knowledge based as well as corpus based similarity is proposed by (Zhu G, 2018). Semantic similarity alone cannot be used at high level of information retrieval which requires combination of relevance and semantic similarity. A hybrid model incorporating both relevance and semantic similarity has been proposed by (Rao, J, et al. 2019). Text mining tools are also successfully used in Bioinformatics domain for extracting protein-protein binding site residues from PubMed abstracts. (Badal, V. D, et al. 2018). Text mining techniques are also used to investigate ability of artificial intelligent agents in answering queries (Hassanzadeh O, et al. 2019)

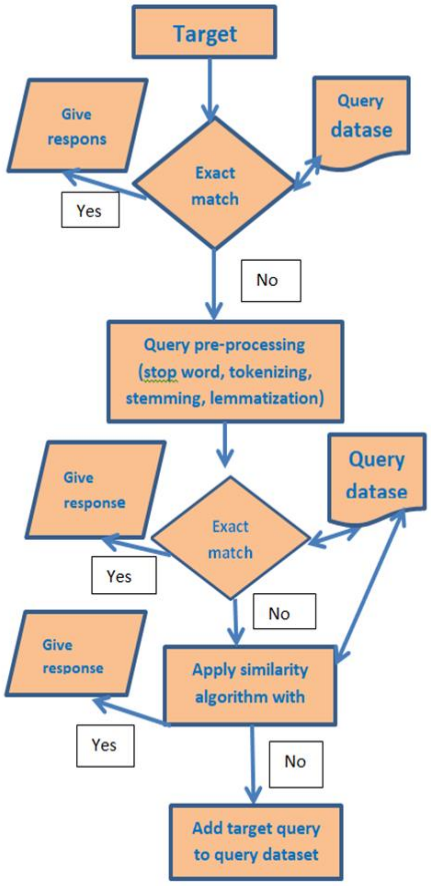
3. Experimental Work

This research work focuses on analysing student's queries using text similarity approach of text mining.

Text similarity algorithms are used to determine how close two queries are. Similarity is measured using both lexical and semantic similarity. Following fig shows the algorithm used in calculating the similarity between queries.

Figure 1

Algorithm for query similarity



To start with first an exact match algorithm is applied on the query without stop word removal. If it matches then similar query's response is displayed as target query response.

Figure 2

Snapshot of code and output for exact match algorithm

```
target_query = "How many seats are reserved for category students for MCA program?"

def exact_match(a, b):

    return (a == b)

for query in dataset:
    print(exact_match(target_query, query), query)
```

```
False Maratha reservation is applicable for MCA program?
False When is the first round of admission ?
False when is spot admission round?
False What was the cut off for first round of admission for MCA program?
False How many seats are reserved for NT-C category students for MCA program?
False How many seats are reserved for NT-B category students for MCA program?
False How many seats are reserved for NT-D category students for MCA program?
False How many seats are reserved for VJ-A category students for MCA program?
False How many seats are reserved for ST category students for MCA program?
False How many seats are reserved for OBC category students for MCA program?
False How many seats are reserved for other university students for MCA program?
False What if category students are not available for admission for MCA Program, how do you fill up seats?
False My results are not declared still will I get admission for MCA Program?
False I am waiting for revaluation result of one of the paper ? Can I appear for admission round for MCA Program?
False I dont have non creamy layer certificate Can I come for admission round of MCA Program?
False Can I do this course externally?
False Can I do this course like partially? Means side by side I can do the job and I can sometime attend the college?
False Is the course full time or part time?
False Can you allow me attend only practicals?
False What is the strength of students in the campus?
False Do I have to stay on campus?
False Is there medical facility available on campus?
```

In exact match methods target query sequence is compared with each dataset query sequence for exact word to word match, if the sequence order differs then the response is not generated. Sentences are not considered equivalent.

E.g. “Process for MCA admission”

“MCA admission process”

Though the meaning of the above queries is same but sequence of words differ so they will not be considered equivalent. As seen in the output no queries are matched with target query.

If the response is not generated for exact match algorithm then approximate sentence matching algorithms are applied on the dataset. In the first phase query pre-processing is carried out where stop words are removed. Stop words removal does not change the meaning of the sentence as they are common words like this, the, are etc.

In the second phase target query and dataset query are split into words called tokens. This process helps in interpreting the meaning of the sentence by investigating the words sequence. In third phase stemming and lemmatization is done. Once the queries are pre- processed similarity algorithms are applied to generate responses. Here Jaccard, Cosine, Euclidean distance, similarity algorithms are used to measure the similarity.

Jaccard similarity index is used for equivalence of the queries. Here lemmatization is already performed on queries as part of pre-processing so directly Jaccard algorithm can be applied. Jaccard similarity is defined as size of intersection of target and dataset queries, divided by size of union of target and dataset queries. Following fig. 3 shows the screen shot of the code and output generated using Jaccard similarity index.

Figure 3

Snapshot of code using Jaccard similarity

```
lemmatizer = nltk.stem.wordnet.WordNetLemmatizer()

def partial_match(x, y):

    pos_x = map(get_wordnet_pos, nltk.pos_tag(tokenize(x)))
    pos_y = map(get_wordnet_pos, nltk.pos_tag(tokenize(y)))
    lemmae_x = [lemmatizer.lemmatize(token.lower().strip(string.punctuation), pos) for token, pos in pos_x \
                if pos == wordnet.NOUN and token.lower().strip(string.punctuation) not in stopwords]
    lemmae_y = [lemmatizer.lemmatize(token.lower().strip(string.punctuation), pos) for token, pos in pos_y \
                if pos == wordnet.NOUN and token.lower().strip(string.punctuation) not in stopwords]

    # Jaccard similarity
    ratio = len(set(lemmae_x).intersection(lemmae_y)) / float(len(set(lemmae_x).union(lemmae_y)))
    return (ratio > 0.66)

for query in dataset:
    lst=[]
    x=(partial_match(target_query, query), query)
    lst.append(x)
    print(lst)
```

```
[(False, 'Maratha reservation is applicable for MCA program?')]
[(False, 'When is the first round of admission ?')]
[(False, 'when is spot admission round?')]
[(False, 'What was the cut off for first round of admission for MCA program?')]
[(True, 'How many seats are reserved for NT-C category students for MCA program?')]
[(True, 'How many seats are reserved for NT-B category students for MCA program?')]
[(True, 'How many seats are reserved for NT-D category students for MCA program?')]
[(True, 'How many seats are reserved for VJ-A category students for MCA program?')]
[(True, 'How many seats are reserved for ST category students for MCA program?')]
[(True, 'How many seats are reserved for OBC category students for MCA program?')]
[(True, 'How many seats are reserved for other university students for MCA program?')]
[(True, 'What if category students are not available for admission for MCA Program, how do you fill up seats?')]
[(False, 'My results are not declared still will I get admission for MCA Program?')]
```

As we see in the above fig.3 after using the Jaccard similarity measure it gave good results. As per the output generated by Jaccard similarities, following table shows the similar queries.

Table -1

Jaccard similarity based similar queries

Target query	
"How many seats are reserved for category students for MCA program?"	
Similar queries:	
1.	How many seats are reserved for NT-C category students for MCA program?
2.	How many seats are reserved for NT-B category students for MCA program?
3.	How many seats are reserved for NT-D category students for MCA program?
4.	How many seats are reserved for VJ-A category students for MCA program?
5.	How many seats are reserved for ST category students for MCA program?
6.	How many seats are reserved for OBC category students for MCA program?
7.	How many seats are reserved for other university students for MCA program?
8.	What if category students are not available for admission for MCA Program, how do you fill up seats?

Next Cosine similarity method is used on the same dataset and target query. Here similarity is measured by the cosine of the angle between two sentences (word vectors) and determines whether two sentences are pointing in approximately the same direction. Following figure shows the part of code and output generated.

Figure 4

Snapshot of code using cosine similarity

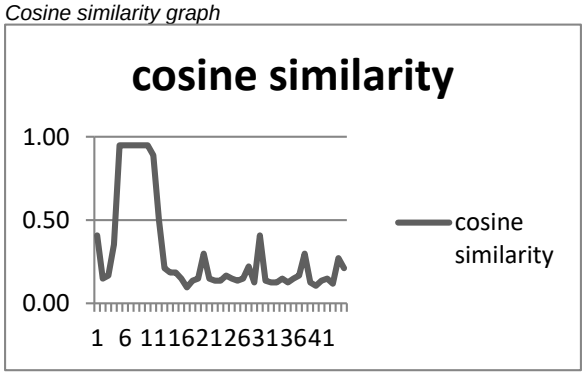
```
a_set = {w for w in a_list if not w in sw}
b_set = {w for w in b_list if not w in sw}


rvector = a_set.union(b_set)
for w in rvector:
    if w in a_set: l1.append(1)
    else: l1.append(0)
    if w in b_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
    c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.408248290463863
similarity:  0.14907119849998599
similarity:  0.16666666666666666
similarity:  0.3535533905932738
similarity:  0.9486832980505138
similarity:  0.9486832980505138
similarity:  0.9486832980505138
similarity:  0.9486832980505138
similarity:  0.9486832980505138
similarity:  0.9486832980505138
similarity:  0.8888888888888888
similarity:  0.502518907629606
similarity:  0.21081851067789195
similarity:  0.1849000654084097
similarity:  0.1849000654084097
similarity:  0.14907119849998599
```

Following graph fig. 5 shows the number of records matching the target query using cosine similarity

Figure 5

Cosine similarity graph



As we see in the above fig.3 after using the Jaccard similarity measure it gave good results. As per the output generated by Cosine similarity matrix, following table -2 shows the similar queries.

Table-2

Training of Neural Networks

Target query	
"How many seats are reserved for category students for MCA program?"	
Similar queries:	
1.	How many seats are reserved for NT-C category students for MCA program?
2.	How many seats are reserved for NT-B category students for MCA program?
3.	How many seats are reserved for NT-D category students for MCA program?
4.	How many seats are reserved for VJ-A category students for MCA program?
5.	How many seats are reserved for ST category students for MCA program?
6.	How many seats are reserved for OBC category students for MCA program?
7.	How many seats are reserved for other university students for MCA program?

Now we apply Euclidean distance similarity on the same dataset. This distance is based on Pythagoras theorem. It the two queries are similar then then Euclidean distance is 0. Following figure shows the part of code and output generated using Euclidean distance measure for similarity.

Figure 6

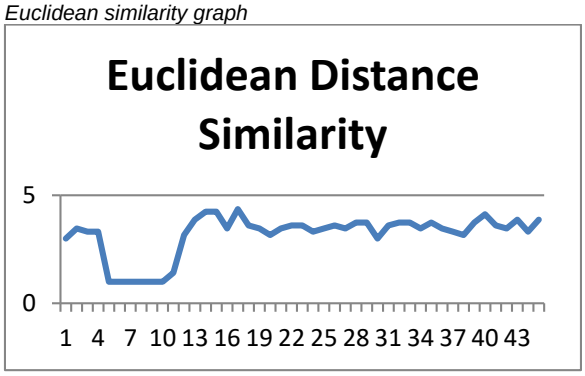
Snapshot of code using Euclidean distance

```
print("Euclidean Distance between source and target is :")
p1=np.array(l1)
p2=np.array(l2)
sum_sq=np.sum(np.square(p1-p2))
print(np.sqrt(sum_sq))
```

```
Distance between source and target is :
3.0
Distance between source and target is :
3.4641016151377544
Distance between source and target is :
3.3166247903554
Distance between source and target is :
3.3166247903554
Distance between source and target is :
1.0
Distance between source and target is :
1.0
Distance between source and target is :
1.0
Distance between source and target is :
1.0
Distance between source and target is :
1.0
Distance between source and target is :
1.0
Distance between source and target is :
1.4142135623730951
Distance between source and target is :
3.1622776601683795
Distance between source and target is :
3.872983346207417
Distance between source and target is :
4.242640687119285
Distance between source and target is :
4.242640687119285
Distance between source and target is :
```

Figure 7

Euclidean similarity graph



The output generated by Euclidean distance similarity is same as cosine similarity as shown in table -2. Here there are six queries with Euclidian distance 1.0 and one with 1.42 score. Figure 7 shows the graphical representation of similarity score using Euclidean distance.

In hunt for better similarity we applied Minkowski distance and Manhattan distance measure to the dataset. Figure 8 shows the source code of Minkowski algorithm and figure 9 shows the output plotted on graph.

Figure 8

Snapshot of code using Minkowski distance

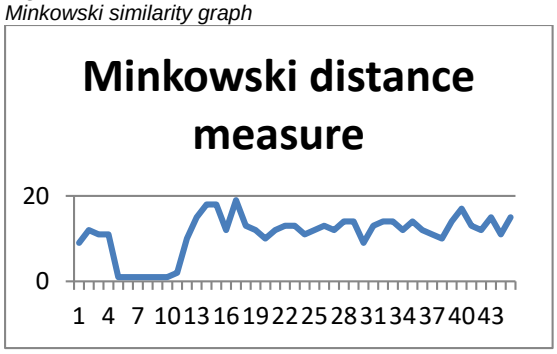
```
def my_p_root(value, root):
    my_root_value = 1 / float(root)
    return round (Decimal(value))

def my_minkowski_distance(x, y, p_value):
    return (my_p_root(sum(pow(abs(a-b), p_value)for a, b in zip(x, y)), p_value))
```

```
    print("Minkowski Distance between source and target is :")

    print(my_minkowski_distance(l1, l2, 3))
```

```
Minkowski Distance between source and target is :
9
Minkowski Distance between source and target is :
12
Minkowski Distance between source and target is :
11
Minkowski Distance between source and target is :
11
Minkowski Distance between source and target is :
1
Minkowski Distance between source and target is :
1
Minkowski Distance between source and target is :
1
Minkowski Distance between source and target is :
1
Minkowski Distance between source and target is :
1
Minkowski Distance between source and target is :
1
Minkowski Distance between source and target is :
2
Minkowski Distance between source and target is :
10
Minkowski Distance between source and target is :
15
```

Figure 9

Minkowski similarity graph



Here Figure-10 shows the source code of Manhattan distance algorithm and figure-11 shows the output plotted on graph. Manhattan distance is the absolute sum of the difference between the x and y coordinates of each data point. Formula implementation is shown in figure-10 below.

Figure 10

Snapshot of code using Manhattan distance

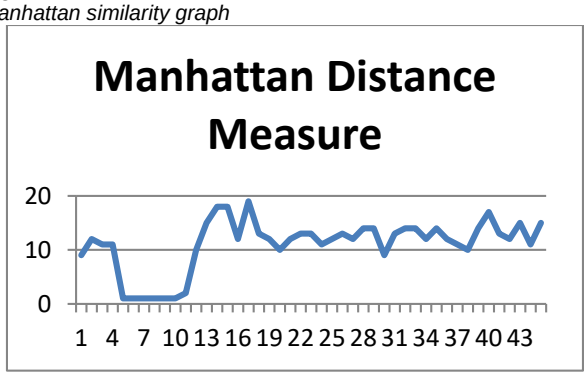
```
def manhattan_dist(x, y):
    return sum(abs(a-b) for a, b in zip(x,y))
```

```
    print("Manhattan between :")
    print(manhattan_dist(l1, l2))
```

```
Manhattan between :
9
Manhattan between :
12
Manhattan between :
11
Manhattan between :
11
Manhattan between :
1
Manhattan between :
1
Manhattan between :
1
Manhattan between :
1
Manhattan between :
1
Manhattan between :
1
Manhattan between :
2
Manhattan between :
10
Manhattan between :
15
Manhattan between :
18
Manhattan between :
```

Figure 11

Manhattan similarity graph



Here in figure-9 and figure-11it is observed that output generated by both Minkowski and Manhattan distance measure is exactly same. The 7th query of the table-2, has similarity measure 2 using both Minkowski as well as Manhattan distance measure. Euclidean distance gave 1.41 as similarity score and Cosine similarity gave 0.88 as similarity index for the 7th query. But Jaccard similarity gave 8 similar queries which is very lenient mapping of query.

It can be observed that all five similarity measures viz. Jaccard, Cosine, Euclidean, Minkowski and Manhattan distance give good results. But Cosine similarity gives more accurate similarity as compared to all other similarity measures in this experiment. The cosine similarity gives better results as, if two queries are quite far by Euclidean distance; there are chances that they may still be closely oriented. Queries with similarity value 1 or near 1 are considered to have high similarity value i.e. If the angle is small then cosine similarity is high.

After applying similarity measures if the query did not have any approximate similarity then it is added to the query dataset along with the intended response.

4. Conclusion

In our study a query corpus is developed with query and related responses. Domain of the research work is limited to student queries related to admission process. In the previous work we could classify the student query in to three categories i.e. admission, placement and fees. This work is in continuation with the previous work. Here corpus is developed with three different FAQ based datasets viz. admission, placement and fees. Once the query is requested by the student it is classified in to its respective category as stated above. After classification, entered query is matched for similarity with its respective query dataset. Here we have used five types of sentence similarity measures to compute the similarity matrix. We could conclude that Jaccard similarity gives lenient approximate matching whereas Cosine, Euclidean, Minkowski and Manhattan distance similarity gives strict approximations results. For student query mapping Cosine similarity gave good results.

Acknowledgement

Above work is carried out under minor research project funded by Shivaji University, Kolhapur under Research Strengthening Scheme.

References

1] Badal, V. D., Kundrotas, P. J., & Vakser, I. A. (2018). Natural language processing in text mining for structural modeling of protein complexes. BMC bioinformatics, 19(1), 84.

2] Hassanzadeh, O., Bhattacharjya, D., Feblowitz, M., Srinivas, K., Perrone, M., Sohrabi, S., & Katz, M. (2019, August).

Answering Binary Causal Questions Through Large-Scale Text Mining: An Evaluation Using Cause-Effect Pairs from Human Experts. In IJCAI (pp. 5003-5009).

3] Holden, E. J., Liu, W., Horrocks, T., Wang, R., Wedge, D., Duuring, P., & Beardsmore, T. (2019). GeoDocA–Fast analysis of geological content in mineral exploration reports: A text mining approach. Ore Geology Reviews, 111, 102919.

4] Kaushik, A., & Naithani, S. (2016). A comprehensive study of text mining approach. International Journal of Computer Science and Network Security (IJCSNS), 16(2), 69.

5] Ramya, R. S., Venugopal, K. R., Iyengar, S. S., & Patnaik, L. M. (2017). Feature extraction and duplicate detection for text mining: A survey. Global Journal of Computer Science and Technology.

6] Ranaei, S., Suominen, A., Porter, A., & Kässi, T. (2019). Application of Text-Analytics in Quantitative Study of Science and Technology. In Springer Handbook of Science and Technology Indicators (pp. 957-982). Springer, Cham.

7] Niemann, H., Moehrle, M. G., & Frischkorn, J. (2017). Use of a new patent text-mining and visualization method for identifying patenting patterns over time: Concept, method and test application. Technological Forecasting and Social Change, 115, 210-220.

8] Rao, J., Liu, L., Tay, Y., Yang, W., Shi, P., & Lin, J. (2019, November). Bridging the gap between relevance matching and semantic matching for short text similarity modeling. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 5373-5384).

9] Salloum, S. A., Al-Emran, M., Monem, A. A., & Shaalan, K. (2018). Using text mining techniques for extracting information from research articles. In Intelligent natural language processing: trends and applications (pp. 373-397). Springer, Cham.

10] Saquicela, V., Baculima, F., Orellana, G., Piedra, N., Orellana, M., & Espinoza, M. (2018). Similarity Detection among Academic Contents through Semantic Technologies and Text Mining. In IWSW (pp. 1-12).

11] Shen, L., Yan, H., Fan, H., Wu, Y., & Zhang, Y. (2017). An integrated system of text mining technique and case-based reasoning (TM-CBR) for supporting green building design. Building and Environment, 124, 388-401.

12] Zhu, G., & Iglesias, C. A. (2018). Exploiting semantic similarity for named entity disambiguation in knowledge graphs. Expert Systems with Applications, 101, 8-24.

About Authors:

Dr. Kavita S. Oza is currently working as senior assistant professor in Computer Science, Shivaji University, Kolhapur. She has 20 years of teaching experience and 10 years of research experience. She has 40+ publications to her credit and 10 reference books. She has successfully guided 3 PhD students. Her research interest is Text Mining, Machine Learning and ICT. She is life member of CSI.

Dr. Poornima G. Naik, received her M.Sc. degree in Physics and Mathematics and Ph.D. degree in physics from Karnataka University, Dharwad. She received MCA degree from IGNOU with first class Distinction. Currently, she is working as Professor in the Department of Computer Studies, CSIBER, Kolhapur. Her areas of interest are network security, soft computing and cloud computing. She has participated in several national and international conferences, authored 20+ books on various cutting edge technologies in IT and has published more than 70 papers in International and national journals of repute. She is a prolific technical writer with excellent communication, analytical and technical skills. She is a recipient of prestigious Dr. APJ Abdul Kalam Life Time Achievement National Award for remarkable achievements in the field of Teaching, Research & Publications awarded by International Institute for Social and Economic Reforms, Bangalore.

Dr. R. K. Kamat is currently I/C Dean Faculty of Science and Technology, Professor in Electronics and heading the Department of Computer Science at Shivaji University, Kolhapur. He is an accomplished University faculty with 25 years of rich teaching and research experience. He is also holding the positions of Director of Innovation, Incubation and Linkages and Director, Internal Quality Assurance Cell at Shivaji University, Kolhapur. Dr. Kamat has to his credit 125+ publications in journals from reputed publishing houses such as IEEE, Elsevier, Springer in addition to 12 reference books from reputed international publishers such as Springer, UK and River Publishers, Netherlands. He has successfully guided 16 Ph.D. students. He has mobilized research grants to the tune of Rs. 24 Cr. from different funding agencies such as UGC, DST and MHRD. He is a Young Scientist awardee of DST under Fast Track Scheme. His latest initiatives are setting up FDC under MHRD PMMMNMTT in Cyber security and Data Sciences, RUSA-Industry sponsored Centre of Excellence in VLSI System Design and coordination of two international projects EQUAMBI and Internationalized Master Degree Education in Nano Electronics under the EU's Erasmus+ scheme