A Novel Video Game Recommender System using Content Based Filtering -Vidya

Krishna Chythanya. N, Krishna Bhargavi. Y, A. B. Rohan

***Abstract:** This work is aimed at building an application that takes input from the user in the form of a few initial games and then proceeds to give recommendations to the user which they will tend to like. Recommendation systems position themselves to fill the gap caused by the presence of voluminous content and the lack of time .This will be done by building a substantial data backend, from which insight similarity will be generated upon processing. Natural Language Processing will be used to gauge the genre, plot and gameplay similarities, Key traits about the games have been extracted and used to fuel the recommendation process. The result was presented to the user in the form of an interactive web application, where they can pick-and-choose their preferred games among the system provided suggestions. The application performance is satisfactory based on Normalized Mutual Information (NMI) score we achieved. The work is applicable to recommend interesting games for the user.*

Keywords: Game, Recommendation, content based filtering, NMI

I. INTRODUCTION

The development of the electronic computer and the stored program computer was the cradle for research in intelligent systems. When humans were able to exploit the vast computing power of these systems, they were lead to wonder whether a machine would ever think and behave like humans did. Thus, the development of intelligent systems started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Recommendation (or recommender) systems are software that provide suggestions to the user about various items which may be of interest to them. The systems suggest the user an item (or multiple), personalized based on their tastes and preferences. They can be contrasted with real world examples, where people generally rely upon the opinion of others before deciding for themselves. For instance, a person who reads books would take into account the choice recommended by a friend for their next book purchase. Recommendation systems position themselves to fill the gap caused by the presence of voluminous content and the lack of time: They aid in guiding the consumers of content towards the better parts. In this work, we created a recommendation engine for video games which bases its recommendations on the key traits about games, obtained using Natural Language Processing techniques. The key techniques which will be used are Keyword Extraction, Bag of Words generation , Cosine Similarity. The paper is organized as, Literature survey in second section followed by Proposed System in the third section, Implementation along with design is discussed at length in the fourth section, results discussed in the fifth section, the sixth section concludes our work with future scope mentioned in the seventh section followed by a bibliography at the end.

Revised Manuscript Received on November 15, 2019

Krishna Chythanya N*, Asst. Prof., CSE, GRIET-Hyderabad-India. kcn_be@rediffmail.com.

Krishna Bhargavi Y, Asst. Prof. CSE, GRIET-Hyderabad-India. kittu.bhargavi@gmail.com

A. B. Rohan, IV Year, CSE, GRIET-Hyderabad-India. rohan.ankb@gmail.com

II. LITERATURE SURVEY

The video game industry leads the entertainment industry with over $116 billion revenue in 2018 alone. Every year, over 9000 games enter the crowded games market. This influx of content ensures that players always have something new to play, but makes the discovery of games harder. A scarcity is created in this abundance.

The existing systems prevalent in the market rely on the principle of collaborative filtering to suggest games. One such system is the KinRate game recommendation service. It works by asking the user for an initial selection of games and proceeds to ask probing questions regarding a user's preferences. The recommendations are then delivered by comparing this model of the user the system has created with models of other, similar users. This entails the collection of massive user data, a major privacy concern. Moreover, such systems are susceptible to user malice as users can exploit the system's functioning.

III. PROPOSED SYSTEM

We utilize the technique of content-based filtering, to make the recommendation process independent of the user as it would solely depend on an individual game's characteristics which are extracted.

Vidya is a game recommendation system that provides recommendations to the user using a content-based filtering approach. The recommendation process will be driven by the characteristic traits of games which will be extracted using NLP techniques. The main features of Vidya are:

Allow the users to select games which they like.

Recommend games to the user based on their preferences.

Users of the system shall be able to view games, select games as per their choice, and receive further game recommendations based on the choices made. If a game is selected, it shall influence the recommendation process, otherwise, it shall have no effect. The system shall support one type of user privilege: Gamer. They shall be able to do the following functions:

- Game selection
- Select game
- Clear game from selection
- Get recommendations
- Start the recommendation process
- View the results of the recommendation process

Game Selection : The user shall be able to select from a pool of games generated and displayed.
The pool shall be fetched from the API backend.
User Choices: The user shall be able to confirm a game as belonging to their selection. The user shall also be able to clear an existing game from their selection.
Recommend Games : The user shall be able to initiate the recommendation process. The games similar to the selection of games shall be displayed to the user.

IV. IMPLEMENTATION

Vidya is a multi-language system, mainly written in Python and JavaScript, using Node.js as the JavaScript runtime. It used React.js as its UI library and Express.js is the web application framework which will be used. MongoDB was used as the primary database. There was modular design throughout the system where different modules communicate through a web API serviced by the Node.js server.
The Python main modules, or packages, made use of in the project are detailed next.
NLTK: Natural Language ToolKit, or NLTK for short, is the de-facto platform to adopt when performing natural language processing tasks using Python.Here, it is used to perform keyword extraction, tf-idf vectorization, etc.
PyMongo : The database used in the project is MongoDB, a NoSQL database which supports reliable and scalable collection of records in a JSON-like format. To connect with a MongoDB instance, a suitable driver is required. PyMongo, module provided by the company behind MongoDB, is used to interface the Python backend with the database.
Scikit Learn :Scikit Learn (sklearn), is an open source, machine learning library in Python. It is built upon other widely popular libraries such as NumPy, Scipy, etc. sklearn contains extensive resources to fulfill the tasks of data preprocessing, feature extraction, etc. It is leveraged here to carry out the generation of bag of words and calculating the cosine similarity.
Requests: Requests is a library for Python which provides powerful wrappers over HTTP requests. The raw data is collected by requesting the IGDB API. The functionality provided by Requests makes this data collection a breeze and saves time by not having to worry about the nitty-gritty details.
Content-Based Filtering: Before we talk about content-based filtering, it is insightful to know about collaborative filtering. In collaborative filtering (or social filtering), the system works by building a model of the user from their past behavior and interactions with the content. The information about the user’s likes, dislikes, etc. is collected and compared with other users of the system. This is used to arrive at potential items the user may be interested in, by observing the patterns of similar users.
Content-based filtering systems do not rely upon user data to come up with recommendations. Instead, they use the unique characteristics of the content to build a network of content similarity. Items which are similar to each other are closer in the network while dissimilar items are farther from each other. The user provides an initial base for recommendation by selecting items of their liking. The system then looks towards items which are similar to the user’s initial selection to give out recommendations. This recommendation range keeps increasing as the user tries out more of the available content.
As there is a lack of reliance on user data, it is necessary to define the important characteristics of the content which act as accurate descriptors of it. Few of these descriptors used in the project include:
Critic and user ratings: There are hundreds of critic ratings, and thousands of user ratings, available for any given game. They provide a general, if not crude, idea about a game’s quality.
Genre classification: Games are available in a variety of genres, such as First Person Shooter (FPS), Action, Role Playing Game (RPG), etc. Users tend to have marked preferences for particular genres.
Developer and publisher profiles: Developers usually have a specific profile where they typically make a particular kind of game. Meanwhile, publishers sometimes specialize in the publishing of a certain subset of games.
Plot and gameplay analyses derived from NLP: Sheer numbers are not enough to properly describe the quality of a game. Contextual information is extracted about a game’s plot and gameplay elements, two factors which immensely influence a game’s ability to captivate the player.
The Similarity Process : The main metric used to give recommendations to the user is what we dub the ‘similarity coefficient’. This similarity coefficient is calculated between every game present in the database and as its name implies, it gives a measure of the similarity between two games. The fields on which this similarness is based upon are the content descriptors of the games discussed earlier. To transform these descriptors into a numeric metric, the technique used is cosine similarity.

Term Frequency - Inverse Document Frequency (tf-idf): For descriptions of content such as plot and gameplay analyses, it is not useful to store the entire blob of text into the bag of words. It diminishes the process’s verity. To circumvent this, a minimized form of representation is required which manages to capture the essence of the information present in the larger text. This is achieved by using the tf-idf technique. Term Frequency - Inverse Document Frequency (tf-idf) is a numeric measure used to score the importance of a word in a document and calculated as per equation (1). It operates on the principle that if a word appears frequently in a document, it is important and should be given a high score. However, if the word appears in too many other documents, it might not uniquely identify the document and is prescribed a lower score.

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \qquad (1)$$

Term Frequency Tf simply calculates the number of times each word appears in each document. To improve the accuracy, common stop words such as “a”, “the”, punctuation marks are excluded and words will be in a single case.

Inverse Document Frequency: While tf gives the importance of any term in a particular document, idf relays the importance of the word when the whole collection of documents is considered. It is calculated as using equation (2) given below.

$$idf(t, D) = log\frac{|D|}{1 + |\{d \in D : t \in d\}|} \qquad (2)$$

$$\vec{a}.\vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = (a^{\to}.b^{\to})/\|a^{\to}\|\|b^{\to}\| \qquad (3)$$

Bag of Words: To gauge the similarity of games, a 'bag of words' is constructed for each game. This metric represents a game's genres, developers, publishers, key plot and gameplay elements. The degree of overlapping of any two games' bag of words gives an indication of the similarity of their creative elements.

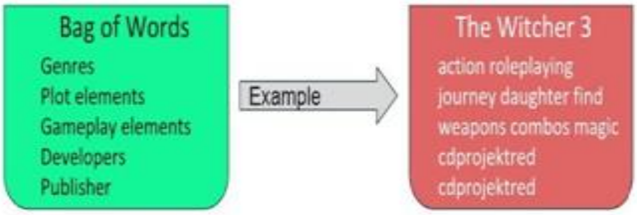


Fig 1: Bag of words

Cosine Similarity : Cosine similarity between two vectors (or two documents on the Vector Space) is the cosine of the angle between them. It determines how similar two documents are, without considering their size (a normalized representation).. In the work, the cosine similarity is calculated between the bag of words of two games. The bag of words is an entity that is made up of accumulating all the content descriptors into a single construct. The formulae for cosine similarity is as shown in equation (3) above.

MERN Stack The foundation for the frontend component of the work was built on the MERN stack. The MERN (MongoDB, Express, React, Node.js) stack is a technology suite used for developing versatile web applications.

The data collections are:

Game Data: The largest collection, it holds the processed data which is obtained cleaning the data obtained from the IGDB API. The IGDB API is a powerful API made available by IGDB.com. It services numerous endpoints to gather information about a vast quantity of games. We use it to collect data about a game's rating, developers, publishers, platforms, genres, and other relevant information.

Bag of Words: After the bag of words is generated from the NLP module, it is stored in this collection. The recommendation engine requests data from this collection when calculating the similarities.

Similarity Data: The recommendation engine calculates similarity coefficients of the games and stores them in this collection.

Game Cards: A sliver of data about a game is stored in the Game Cards collection. It is mainly used by the frontend to display content to the user.

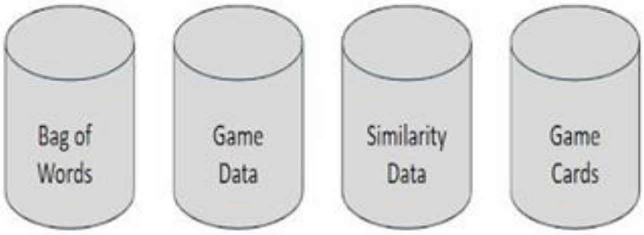
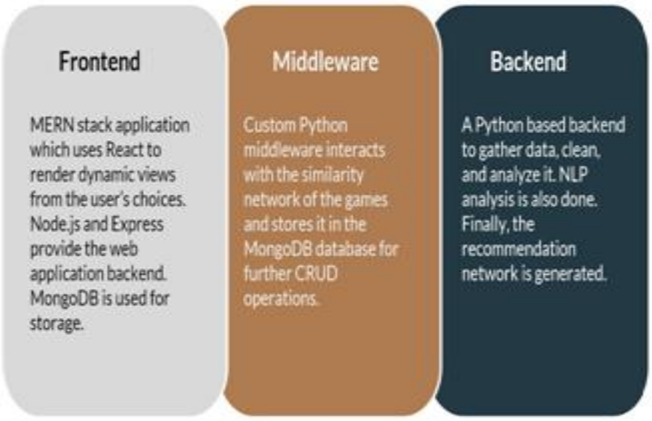


Fig 1: The collections used

System Architecture: The system architecture is built up in such a way that the functioning can be split up into three main phases. There is the data processing phase, taken up by the backend, where data is gathered, cleaned, and analyzed. In the next phase the recommendation network is generated. The backend and the middleware are both involved in the generation of the recommendation network. The final phase is where the user interacts with the application. The frontend is responsible for this phase.



MERN stack application which uses React to render dynamic views from the user's choices. Node.js and Express provide the web application backend. MongoDB is used for storage.

Custom Python middleware interacts with the similarity network of the games and stores it in the MongoDB database for further CRUD operations.

A Python based backend to gather data, clean, and analyze it. NLP analysis is also done. Finally, the recommendation network is generated.

Fig 2:The System Architecture

A common interface between these three phases is the MongoDB database which stores data required by different components. at different points in the process.

Data Processing Module

The work makes use of vast amounts of data. This data had to be gathered, cleaned, and organized into a structured format to enable smooth usage further down the project pipeline. This was done by the data processing module. The Python based module makes calls to the IGDB API and fetches information about the games. This raw data is then given structure and transformed into a Python dictionary for future usage. The data received from the API call is comparatively large and the task of making the content concise is handled by this module. While performing this task, the data is also sanitized and invalid data values are normalized

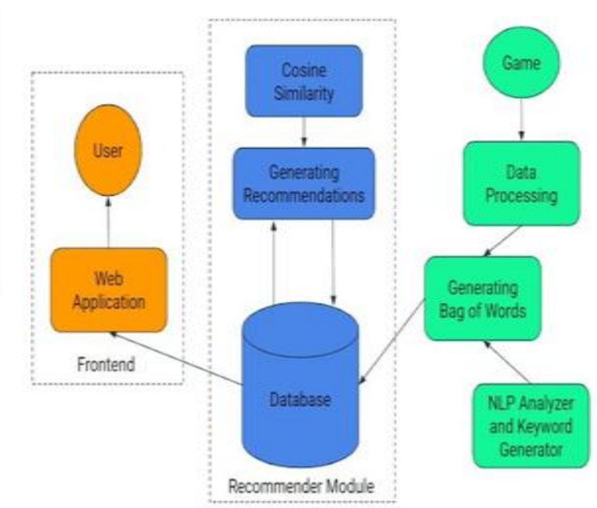


Fig 4: Flow diagram

NLP Analyzer:

Bag of Words Generator: This module performs the key step of generating the bag of words on which the further processes depend upon. The bag of words generator takes the structured game data from the data processing module and the newly generated keywords about the game from the NLP analyzer and keyword generator module and merges them into a unigram bag of words. This is a crucial step as much of the further recommendation process relies on these bags of words.

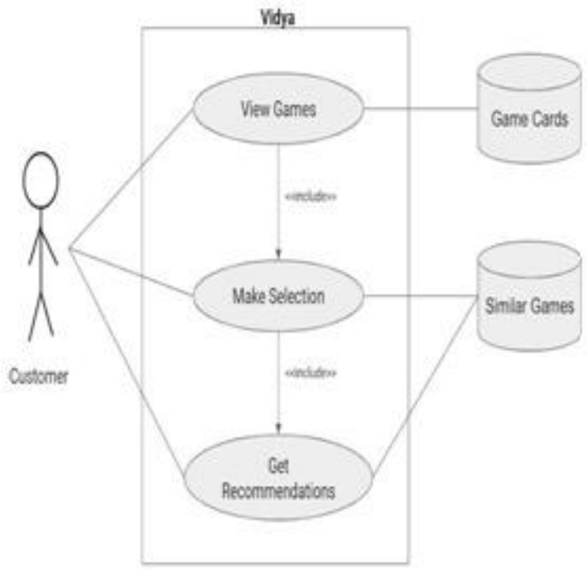


Fig 5:Use case diagram of application

Recommender Module: The core function of determining game similarity on which the recommendation system works is done by the recommender module.

The key component of the recommender module is the cosine similarity calculation. Two individual games are selected and represented as their bags of words. The cosine similarity calculated between these two bags of words is the measure of the overall similarity between the two games. The similarity coefficient is a value ranging from 0 to 1; if the value is closer to 0, then the games are widely dissimilar, whereas if the value is close to 1, then the games are extremely similar. After calculating this similarity coefficient, the module is responsible for building the wider recommendation network and storing it in the database.

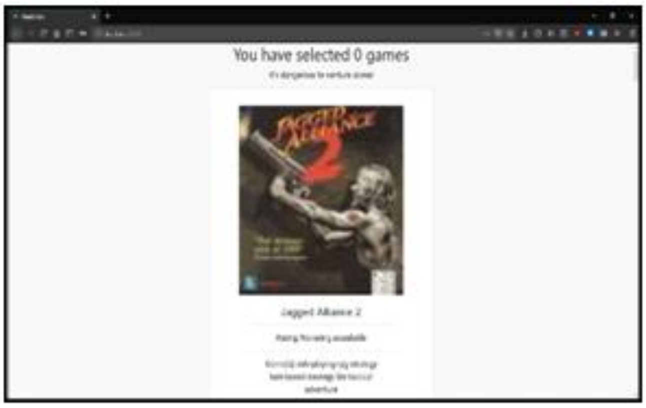
V. RESULTS DISCUSSION

The initial splash screen and the game selection screen are shown in figures Fig 6 & Fig.7 in end.. The user goes through the selection and makes choices as shown in Fig 8. At end.

After the user has finished selecting, they are shown the recommendations as shown in Fig 9. At end. The NMI score for different runs of the application is shown below in Figure 11. Below. It was considered for 10 runs and the average NMI score was observed to be 0.714 i.e. 71.4% which is a reasonably acceptable value. The graph of average NMI score is as shown below Fig 10.

Crawl

Fig 6: Initial Splash screen of Application.


You have selected 0 games

Jagged Alliance 2

Fig 7: Game selection Screen

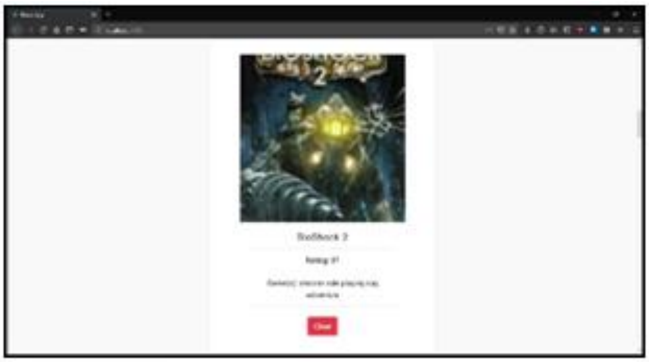
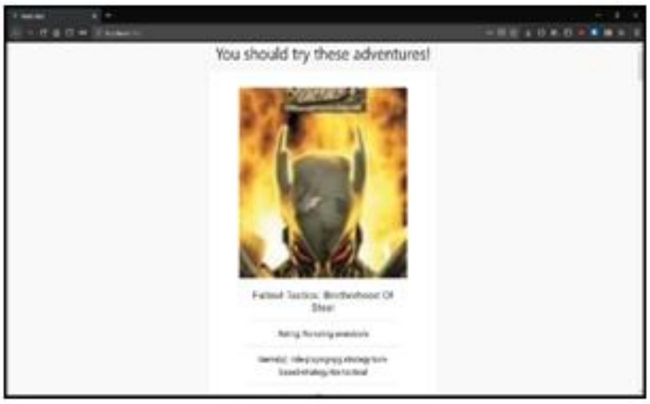

BioShock 2

Fig 8: User Choice done



You should try these adventures!

Fallout Tactics: Brotherhood Of Steel

Fig 9: Recommended Games by the application

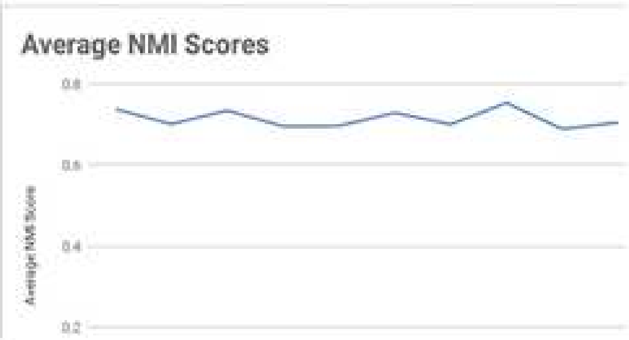
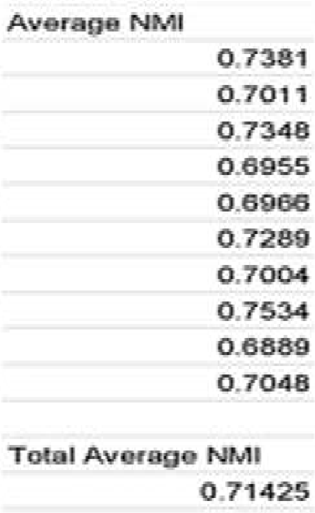


Fig 10. Graph of Average NMI Scores.



Average NMI
0.7381
0.7011
0.7348
0.6955
0.6966
0.7289
0.7004
0.7534
0.6889
0.7048
Total Average NMI
0.71425

Figure 11: Average NMI scores of 10 running of the application.

VI. CONCLUSION:

The primary goal of the research was to allow the users to discover rich experiences which they would otherwise be blind to, in massive sea of games available online. Another goal of the project was to buck the trend of collaborative filtering recommender systems employed in the domain of video games. By opting for a content-based filtering route, we envisioned that the recommendations provided by our system would be free of the user bias rampant in the gaming community. This user bias manifests itself in both forms: unwavering adulation and unflinching scorn. We took a base user for comparison and jotted down their video game likes and dislikes. The KinRate system was used as the standard for comparison. The user profile was fed into the KinRate system which churned out its recommendations by analyzing, and comparing, the user's patterns with other users. Meanwhile, our system worked by giving the user a choice of initial seed games and then meted out recommendations based on the user's choices. To contrast the two systems, we used the array inversion approach, where the elements are ranked according to their indexes and a greater inversion count implies a greater deviation. The initial game provided to both systems was 'Baldur's Gate'. When the KinRate system was given a dearth of data to work off of, the system was wildly inaccurate (most of the suggestions were sports games, not turn-based-RPGS). Meanwhile Vidya recommended games such as 'The Elder Scrolls', and 'Baldur's Gate II'. By providing for a system which negates this extreme behavior of the gaming community, we hope to have made a contribution to wider discussion of objective analysis of video games.

FUTURE SCOPE

It might be ironic to suggest the inclusion of collaborative filtering as an improvement to a project whose main goal was to provide an alternative to such methods, however, as the application grows to have a sizeable user base, these users will be generating treasure trove of data points. A collaborative inclusion would entail the proper collection of this data from the users (whilst respecting their privacy) and analyzing it to enhance the existing base system.

The decision to pick React as our frontend library has already poised us to expand the project into a Progressive Web App (PWA). By redesigning few of the existing parts of the system to be more mobile-friendly, a PWA version of the application will increase the number of users who get to interact with it.

BIBLIOGRAPHY

1. Hossein Jafarkarimil, Alex Tze Hiang Sim, and Robab Saadatdoost, "A Naive Recommendation Model for Large Databases", International Journal of Information and Education Technology, vol. 2, no.3, pp. 216-219, June 2012 (ISSN: 2010-3689).
2. Linden D.Gregory, "Collaborative Recommendations Using Item-to-Item Similarity Mappings", U.S., Patent 6,266,649, issued July 24, 2001.
3. Mooney, R. J., and Roy, L. 2000. "Content-based book recommending using learning for text categorization" in Proceedings of the Fifth ACM Conference on Digital Libraries, 195–204.
4. Michael J. Pazzani and Daniel Billsus, "Content-based recommendation systems", The Adaptive Web: Methods and Strategies of Web Personalization . Volume 4321 of Lecture Notes in Computer Science, pp. 325-341, 2007.
5. Michael Fleischman and Eduard Hovy, "Recommendations Without User Preferences: A Natural Language Processing Approach ", Marina del Rey, CA, 2003.
6. Prem Melville and Vikas Sindhwani, "Recommender Systems Handbook", Encyclopedia of Machine Learning, ch. 38, pp. 1-9, 2010.
7. Michael Howe, "Pandora's Music Recommender", Washington, 2007.
8. James LeDoux, "Building a Content-Based Recommender System for Books: Using Natural Language Processing to Understand Literary Preference", https://jamesrledoux.com/projects/gutenberg-recommender-system/.
9. Reuters, "Investing in the Soaring Popularity ofGaming",https://www.reuters.com/sponsored/article/popularity-of-gaming

AUTHORS PROFILE

Mr. Krishna Chythanya Nagaraju, having B.E.(Comp.Tech),M.Tech(C.S.E.) degrees, has an overall engineering subjects' teaching experience of more than 15 years, which also comprise of around 6 years research experience. He successfully held many academic administrative positions at different colleges and is currently pursuing PhD in the area of Neural Networks and Software Engineering at O.U.-Hyd. He has many research papers published in various National/International Journals/Conferences to his credit. He is open for collaborative research work with like minded people. His areas of interests besides PhD domain includes Information Security, Operating Systems, Data Structures.

Y.Krishna Bhargavi is presently engaged in pursuit of Ph.D in Big Data Analytics and Cloud Computing, registered at JNTUK, Kakinada. Prior to joining Ph.D, she had earned Bachelors of Technology in Information Technology and Masters of Technology in Software Engineering. She published papers in various International Journals and Conferences**.**

A.B. Rohan is pursuing his 4th year of B.Tech in Computer Science and Engineering at GRIET, Hyderabad. He previously interned at Microsoft IDC, Hyderabad over the course of Summer 2019. His areas of interest in research are Natural Language Processing (NLP), Image Processing, and Deep Learning techniques such as GANs and CNNs.