An Extensible Parsing Pipeline for Unstructured Data Processing

Shubham Jain*, Amy de Buitléir†, and Enda Fallon*
*Software Research Institute, Athlone Institute of Technology, Athlone, Ireland
†Network Management Lab, Ericsson, Athlone, Ireland
sjain@ait.ie, amy.de.buitleir@ericsson.com, efallon@ait.ie

***Abstract*—Network monitoring and diagnostics systems depict the running system's state and generate enormous amounts of unstructured data through log files, print statements, and other reports. It is not feasible to manually analyze all these files due to limited resources and the need to develop custom parsers to convert unstructured data into desirable file formats. Prior research focuses on rule-based and relationship-based parsing methods to parse unstructured data into structured file formats; these methods are labor-intensive and need large annotated datasets. This paper presents an unsupervised text processing pipeline that analyses such text files, removes extraneous information, identifies tabular components, and parses them into a structured file format. The proposed approach is resilient to changes in the data structure, does not require training data, and is domain-independent. We experiment and compare topic modeling and clustering approaches to verify the accuracy of the proposed technique. Our findings indicate that combining similarity and clustering algorithms to identify data components had better accuracy than topic modeling.**

***Index Terms*—Unsupervised Data Mining, Information Extraction, Clustering, Topic Modeling**

I. Introduction

Network monitoring and diagnostics have been a topic of interest in recent years. Due to the growth of telecommunications networks and the advancement in 5th Generation (5G) technologies and virtualization, a large volume of data is generated every second. This data contains abundant insights on various system components, and can be used to achieve higher data rates, low latency, reliability, and network optimization. However, analysis is often time-consuming due to incoming data; it is not feasible to scan all messages to identify systemic issues and patterns [1]. Parsing the data presents several challenges. The data is presented in various custom text formats; these formats are subject to change with software updates. The data typically contains extraneous information that needs to be filtered, and then useful information needs to be parsed and processed into a structured file-format. Manually identifying and extracting information from data streams used for analysis is a labor-intensive process and requires domain expertise. General-purpose parsers can struggle with the domain-specific technical language, yet developing and maintaining custom parsers for each format is time-consuming.

A telecommunications domain has various components that generate data streams with a wide variety of custom text formats subject to change; developing and maintaining parsers for each format is time-consuming. General-purpose parsers can struggle with the domain-specific technical language. Recently, some automatic methods have become available that extract information based on the structure of the text; however, they are domain-specific and can fail to parse text that varies in structure or formatting. Thus, there is a need for an automatic domain-independent method to extract relevant information from the free-form text to support the creation of a knowledge base [2].

Because of these difficulties, typically only a small fraction of the available data is analyzed; it is not easy to extract the data to create a knowledge base. Furthermore, large training sets are needed for training, and they must be updated periodically to adapt to the latest networking policies and standards. Text classification using natural language techniques requires a dictionary of all the free form corpus elements. They contain extraneous text, nested technical words, non-standard abbreviations, or grammatically ambiguous text. Creating a dictionary for a large scale domain that has an extensive array of components developed by various companies is not feasible [3]. A text parser is needed that is 1) Resilient to variation in data formats, 2) Identify relevant and extraneous information without manual handling, 3) Does not require labeled training data, 4) Does not require human-intervention and 5) Does not require a dictionary or meta-data to parse different table components such as rows, columns, and header. A solution that does not suffer from these drawbacks is needed.

In this paper, we present an extensible text processing pipeline that analyses various data files with uneven character distribution and large variance in structure across different data sources, identifies the extraneous information blocks in the structure, and removes them, and extracts tabular components to later parse them into a structured data format. The processed data can be made available for exploratory data analysis, anomaly detection [4], incident management, autonomous problem validation [5], root cause analysis, and many more.

This paper is organized as follows: We describe existing

approaches in §II and present our methodology in §III. We also experiment with our method with other approaches and present the results in §IV, and we conclude in §V.

II. Related Work

In this section, we discuss the different approaches for data extraction and parsing from free-form data. We divide them into relationship-based and rule-based approaches. We further discuss the characteristics of data that are challenges for parsing.

A. Rule-based Extraction

Rule-based extraction used pattern mining techniques and heuristics-based analysis. Such approaches are invariant to data structure changes and focus on extracting a structure inside free form text. Cleaning data and extracting relevant information requires domain knowledge, and the process is time-intensive. As the information has many non-linguistic technical terminologies, acronyms, and abbreviations that would not be found in a general-purpose dictionary, the authors of [6] used an agent-based approach. There were multiple agents in the pipeline that filtered data based on protocols set by each agent. Due to each agent's handcrafted rules, the approach is not viable to variance in data. The approach does not satisfy requirement 3 as it falls short on processing files with variance in structure.

In contrast, the authors of [7] implemented a pattern-based extraction method. They identified templates in a log file using regular expressions and generated a dictionary of those templates. The constant part is familiar characters that repeat for each log in an iteration. Variable parts that are unique in log lines were separated from the log file. The technique is data-driven and widely used for the extraction of information from the text. The algorithm iterates through the data and generates information on word frequency and distribution using association rule mining [8], which detects correlated feature patterns that were used to cluster log templates. This method identified different structures but fell short on requirements 2 and 4, identifying extraneous information and the need for a domain expert to identify different clusters later parsed using regular expression templates.

The authors of [9] clustered log messages using word frequency. They generate word-score pairs for each word in a corpus and create templates after iterating through lines and clustering words with similar scores. The method works well for logs with a relation between different file components such as key-value pairs, similar whitespace distribution inside a file. However, requirement 4 is not fulfilled as the processing of files requires a domain expert to identify relevant information. Thus, manual handling of the files needs to be done, which is not feasible for a large volume of data.

In summary, rule-based extraction focuses on extracting a pattern from free-form text. They are not generic enough to apply to different structures across a large scale system due to the need of a domain expert.

B. Relationship-based Extraction

A large scale system generates text files in a wide variety of formats from different components that have no model defined. In relationship-based extraction, the stream of data inside a text file can be analyzed to categorize it into various structural components. The authors of [10] classified free-form data into different categories using the Support Vector Classifier. They applied Conditional Random Fields (CRFs) to extract entities from the unstructured text, and then categorized the entities as text, features, or both. CRFs are useful for extracting a sequence of characters that resemble a pattern. One drawback of this approach was the need for annotated training data which is requirement 3. Also, any update in the system would require re-training the model that classified words in the unstructured text file.

This approach of classifying text entities was further optimized by the authors of [11] who calculated the Term Frequency / Inverse Document Frequency (TF-IDF) [12], which is the frequency of words in the unstructured text. This yielded better results in identifying different components as entities than n-grams that compute the contiguous sequence of characters in a word and time statistics due to weights in word distribution. The same authors also concluded using F1-score [13] that the Self Organizing Feature Maps (SOFM) variant with TF-IDF statistics as features [14] for classifying word components performed comparably with clustering algorithms such as K-means [15]. SOFM groups several clusters closer to each other by analyzing the distance between the points in bi-dimensional space. This solved the problem of analyzing a high volume of data in one iteration. However, the approach didn't fulfil requirement 2 as it required the unstructured file to be cleaned before it was analyzed, and the process was manual and time-consuming.

The rule-based and relationship-based extraction methods fall short on addressing extraneous information filtering and extraction, which is a crucial step in processing a free form text file. An extensive literature review of various techniques is also presented in [3]. However, the approaches are sensitive to changes in the file structure and require manual cleaning of a file for automatic parsing. None of these approaches satisfy all of the requirements we identified in §I.

III. Methodology

In this section, we describe the methodology of our parsing pipeline. We present, evaluate, and compare two approaches, 1) Similarity-vector and Clustering-based, and 2) Topic Modeling-based. A simple illustration of our unstructured data parsing is shown in Fig.1. A text file is used as an input to the

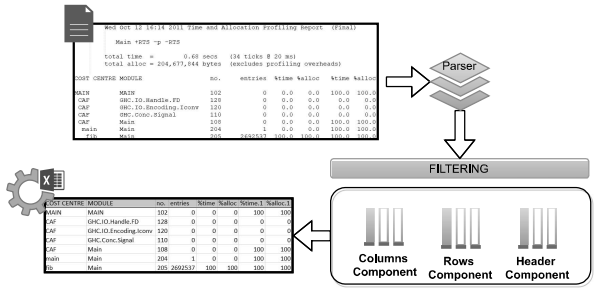

Fig. 1: An Illustrative Example of Unstructured Data Parsing

parser; the next step is identifying useful information from the text that can be used for exploratory data analysis. We remove information such as timestamp, metadata, and any information that is not useful. The filtered file is then analyzed to identify different tabular components such as header rows, data rows, and columns. The analyzed file is then parsed into a structured format and can be used for various machine learning applications.

A. Loading and Pre-processing

For both the approaches, we load the console output generated from network devices to a raw text file with UTF-8 encoding and identify phrases with consecutive special characters. The input text might have separators that are any consecutive special characters such as dashes between table headers and table data. Typically consecutive special characters or whitespaces are used as separators. However, separators are not always present. To make our system more generic, we use regular expressions to recognize separators and remove them from our input.

B. Noise Elimination

After processing the file, we create a similarity matrix using Levenshtein distance [16] that calculates the number of string manipulations required for them to be similar. The resulting output is a score ranging from [0-1] and presents the similarity of two lines in our input file. Equation 1 is the formula to calculate Levenshtein distance for two strings x and y is given in Equation 1

$$lev_{x,y} = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} lev_{x,y}(i-1,j) = 1 \\ lev_{x,y}(i,j-1) = 1 \\ lev_{x,y}(i-1,j-1) = 1 \end{cases} & otherwise \end{cases} \tag{1}$$

We use this similarity matrix to cluster the lines of text using an Agglomerative clustering [17]. This clustering technique identifies the two closest lines and merges them to form a cluster, repeating this all the lines are merged into a single cluster. The distance between consecutive pairs of nearest lines is recorded to generate a dendrogram that classifies the input file into noise and data. We described this method in detail in [18].

We also implemented a novel method using topic modeling algorithms. We pass the input file and split each line with whitespace as a separator to generate a token. We then use the tokens to transform the contiguous sequence of texts using n-grams [19]; , grouping words into bi-grams and tri-grams and creating a dictionary of words. We make a topic model using Gensim [20], a python library to analyze statistical semantics in a file. Linear Discriminant Analysis(LDA) [21] is a classification algorithm that finds between-class variance by calculating the distance between the mean of classes given in Equation 2;

$$b = \sum_{i=1}^{g} N_i (x_i - x)(x_i - x)^T ; g = group, T = threshold \tag{2}$$

and within-class variance w which is the distance between the mean and sample of each class given in Equation 3;

$$w = \sum_{i=1}^{g} (N_i - 1) S_i \quad ; g = group \tag{3}$$

to construct a lower dimensional space P that maximizes b and minimizes w given in Equation 4.

$$P_{lda} = argmax \frac{|P^T b P|}{|P^T w P|} \quad ; T = threshold \tag{4}$$

LDA was used to create a topic model. The number of topics k is kept as 2, so that the model only classifies lines into two topics; Noise(N) and Data(D). However, the coherence score, an evaluation metric to calculate the topic model's efficiency, was less as the value of k was preset.

C. Data Parsing

We further use the classified file with noise and data to identify table components. We analyze data rows using white space distribution and index positions of the words. Our algorithm then identifies columns and header components and transforms them into rows presented in Algorithm 1. We analyze lines and calculate a similarity score and meta score, which is the word's size and the front and back whitespaces. We use these scores to calculate mean and then split words in the line and append them to a column. We further analyze the maximum similarity score in the lines and label that as our header line.

The topic modeling method does component extraction by changing the number of topics k from 2 to 3 for our LDA topic model. The model generates three topics that are noise, data, and header. We filter out the noise label and split columns from

Algorithm 1 Component Extraction

```
procedure IDENTIFYCOMPONENT
    similarityAlgorithm ← Levenshtein
    token ← [word]
    df ← Read Data Rows
    meta ← [startIndex, endIndex, whitespaces]
    scoreList,metaList ← []
    for df.iterrows() do
        for df_i.iterrows() do
            x_i ← df[data].Read Line 1
            meta ← x_i end − x_i start + x_i whitespaces)
            y_{i+1} ← df[data].Read Line 2
            score ← callAlgorithm(x_i, y_i)
            ScoreList ← append(score)
            metaList ← append(meta)
            i ← i + 1
        end for
        header_i ← max(scoreList)
        meanScore ← mean[scoreList,metaList]
    end for
    for df.iterrows() do
        x_i ← df[data].Read Line 1
        for i:metaList,scoreList do
            if x_i metaScore − x_{i+1} metaScore = 0 then
                if x_i metaScore − x_i meanScore < 1 then
                    row ← append(word)
                end if
            end if
        end for
        df_i ← append(row_{header_i})
        if i != header_i ndex then
            df_i ← append(row_i)
        end if
    end for
end procedure
```

Algorithm 2 Topic Label for Each Line

```
procedure FINDTOPIC
    file ← read file
    topicList ← []
    words ← file.gensim.preprocess                ▷ Tokenize
    bigram ← gensim.phraser(words)
    trigram ← words.gensim.phraser(bigram)
    dictionary ← corpora.dictionary(trigrams)
    for i:words do
        corpus ← id2word.doc2bow(i)               ▷ Create Corpus
    end for
    ldamodel ← [corpus, topic ← 3,
                    randomstate ← 100, alpha ← true]
    row ← sorted(words)
    for i, row in file do
        line ← file.read
        topic ← ldamodel.showtopic(line)
        topicList ← append(topic)
    end for
end procedure
```

Algorithm 3 Find Optimal Topics

```
procedure OPTIMALTOPIC
    coherence ← []
    modelList ← []
    for numtopics:range(start,limit,step) do
        model ← gensim.LdaMallet
        modelList ← append(model)
        coherencemodel ← ConherenceModel(model,
                            text, dictionary,coherence)
        coherence ← append(coherencemodel)
    end for
    modellist, coherence ← ComputeCoherenceVal
    newmodel ← optimalmodel.showtopics
end procedure
```

data and header lines using whitespace distribution and meta scores similar to the previous approach. We then transpose the generated rows and columns of data and create another LDA topic model. As each input sample can have multiple columns, we wanted to treat each column as one topic. So we analyze the best value *k* for each input sample. The algorithm to analyze calculate the optimal topics is given in Algorithm 3. We use the *compute coherence value* method in Gensim that computes all *k* values and provides model coherence scores. After choosing the best value *k*, we use that model to identify columns and then append them to rows to create a structured file format. The algorithm is given in Algorithm 2

Both the methods work well for identifying extraneous information, and the results are discussed in §IV.

IV. Evaluation and Discussion

In this section, we present the results of both the methodologies discussed in the earlier section. We identify the most accurate approach and present their accuracies using a confusion matrix. Furthermore, we discuss dataset characteristics and evaluation metrics.

A. Dataset

We evaluated our pipeline on a sample dataset provided by Ericsson. The details are hidden as it contains proprietary information. Understanding different components of data in an automated way to remove noise and make it available for parsing followed by analysis is a significant upgrade in terms

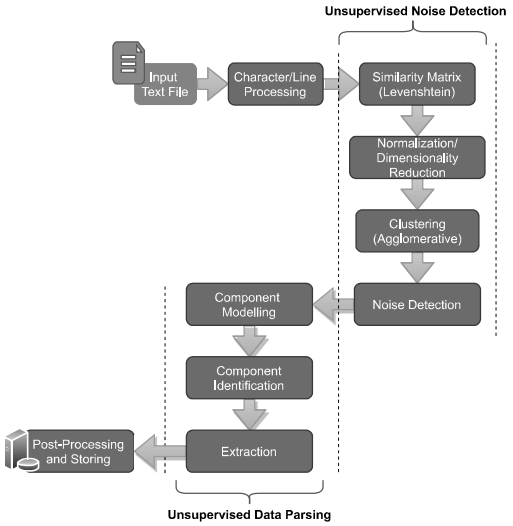

Fig. 2: An Extensible Pipeline for Unstructured Data Parsing

of cost and resources compared to the analysis done by domain experts.

We also use an open-source dataset from ntc-templates [22] with 200 samples from different networking vendors. Each text file contains unstructured lines classified into noise, data, header, and columns. As the sample was a raw text file, there was no label provided for evaluation. We manually created a new dataset with 50 samples from ntc-templates and sample Ericsson dataset. We annotated them with the expected label value. We used various samples from syslogs that generated an output ranging from [200-500] lines. Our manually created output datasets contained four expected label values for each line; Noise lines (N), Header/Data lines (R), Data lines (D), Header line (H). The total number of lines in each sample ranges from [50-100] with different characteristics in data such as missing values in data lines, uneven distribution of column words in data rows, variance in structure using special characters to form table lines, samples that contain less more noise lines and samples that have less or no noise lines.

B. Evaluation

F1-score is the evaluation metric that we use for calculating accuracy. We calculate precision given in Equation 5 by computing the ratio of correct positive predictions with all positive predictions for each label ,

$$precision = \frac{TP}{TP + FP} \tag{5}$$

and recall given in Equation 6 by calculating the ratio of positive predictions labels to all positive and negative predictions for each label.

$$recall = \frac{TP}{TP + FN} \tag{6}$$

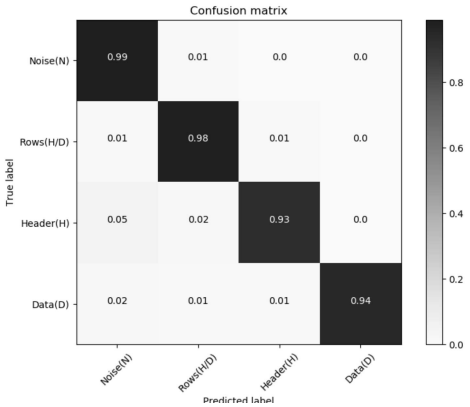

Fig. 3: Accuracy of Similarity Vector-based Parsing

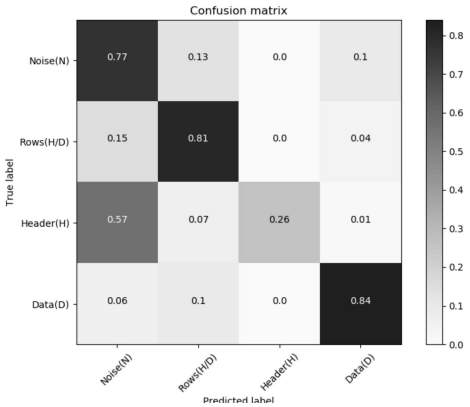

Fig. 4: Accuracy of Topic Modeling-based Parsing

We then calculate F1-score given in Equation 7 by calculating the harmonic mean of precision and recall and then present them in a confusion matrix.

$$F = 2.\frac{precision\ .\ recall}{precision + recall} \tag{7}$$

C. Results

The accuracy of the method that uses Levenshtein distance to calculate similarity and Agglomerative clustering for filtering and parsing is presented in Fig.3. This method identifies noise, relevant rows, header, and data with higher accuracy, as seen in the figure. Identifying header components from data is confusing for the algorithm for some structures where the data contains similar keywords. On the other hand, the accuracy of the Topic Modeling method is presented in Fig.4. As seen in the figure, this method works well to segregate relevant data from noise in most cases, but it fails to segregate header components from relevant data rows.

Method 1 is illustrated in Fig.2, the pipeline contains two significant phases, an unsupervised noise detection that filters

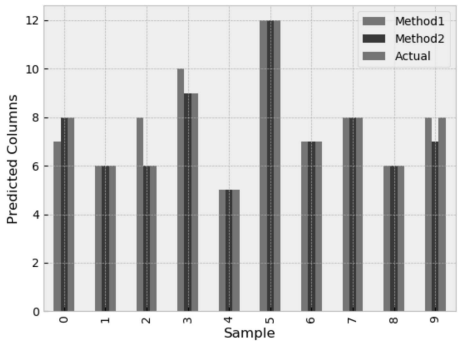

Fig. 5: Column Extraction Accuracy

extraneous information, and a data parsing component that identifies tabular components and extracts them into a structured file format. We used 10 data samples with the highest column structure variation to calculate column component extraction accuracy by method one and method 2. As seen in Fig.5, we plotted the predicted columns by each method and actual columns for ten samples. The results suggest that both the approaches worked well for column component extraction; however, classification of columns as topics provided better predictions than finding similarity of lines and calculating the mean score of whitespace distributions. Thus, method 1 has more accuracy in eliminating extraneous information and detecting relevant rows and header, while method 2 has better accuracy in extracting column components. Method 1 is illustrated in Fig.2, the pipeline contains two significant phases, an unsupervised noise detection that filters extraneous information, and a data parsing component that identifies tabular components and extracts them into a structured file format.

V. Conclusions

This research describes a text processing pipeline that analyses unstructured data, filters extraneous information, identifies data, header, and column components, and parses them into a structured file format. The pipeline is resilient to a high degree of variation structures, does not require annotated training data, and extensible for data generated by an extensive array of components in the network without the need for metadata or dictionary. The approach minimizes the efforts needed in cleaning and data wrangling and does not require human intervention. The pipeline's output can be utilized for data exploration and machine learning analysis, yielding insights into the system that generated the data.

Acknowledgments

This work is funded by Irish Research Council Enterprise Partnership Scheme Postgraduate Scholarship 2020 under Project EPSPG/2020/7.

References

[1] R. Bhowmik and A. Akyamac, "Domain-independent automated processing of free-form text data in telecom," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1841–1849.

[2] W. Paik, S. Yilmazel, E. Brown, M. Poulin, S. Dubon, and C. Amice, "Applying natural language processing (nlp) based metadata extraction to automatically acquire user preferences," in *Proceedings of the 1st international conference on Knowledge capture*. ACM, 2001, pp. 116–122.

[3] S. Jain, A. de Buitléir, and E. Fallon, "A review of unstructured data analysis and parsing methods," in *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2020, pp. 164–169. [Online]. Available: https://doi.org/10.1109/ESCI48226.2020.9167588

[4] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, oct 2016.

[5] W. Shang, Z. M. Jiang, H. Hemmati, B. Adams, A. E. Hassan, and P. Martin, "Assisting developers of big data analytics applications when deploying on hadoop clouds," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 402–411.

[6] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, dec 2009.

[7] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003)(IEEE Cat. No. 03EX764)*. IEEE, 2003, pp. 119–126.

[8] C. Zhang and S. Zhang, *Association Rule Mining: Models and Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2002.

[9] L. Tang, T. Li, and C.-S. Perng, "Logsig: Generating system events from raw textual logs," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 785–794.

[10] C. Suh-Lee, "Mining unstructured log messages for security threat detection," 2016.

[11] W. Li, "Automatic log analysis using machine learning: awesome automatic log analysis version 2.0," 2013.

[12] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 2004.

[13] T. Joachims, "A support vector method for multivariate performance measures," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 377–384.

[14] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[15] K. Krishna and N. M. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems Man And Cybernetics-Part B: Cybernetics*, vol. 29, no. 3, pp. 433–439, 1999.

[16] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb 1966.

[17] M. L. Zepeda-Mendoza and O. Resendis-Antonio, *Hierarchical Agglomerative Clustering*. New York, NY: Springer New York, 2013, pp. 886–887. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_1371

[18] S. Jain, A. de Buitléir, and E. Fallon, "Unsupervised noise detection in unstructured data for automatic parsing," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–5. [Online]. Available: https://doi.org/10.23919/CNSM50824.2020.9269096

[19] D. Schmidt and C. Heckendorf, "ngram: Fast n-gram tokenization," 2017, R package version 3.0.4. [Online]. Available: https://cran.r-project.org/package=ngram

[20] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[21] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[22] Networktocode, "networktocode/ntc-templates." [Online]. Available: https://github.com/networktocode/ntc-templates

Shubham Jain is a Software Researcher and Lecturer with a demonstrated history of working in the Software Development and Research. His research interest focuses on Data Science, Machine Learning, Artificial Intelligence, and Software Development applications. Strong engineering professional, pursuing Doctor of Philosophy - Ph.D. focused on Computer Science from Athlone Institute of Technology. Lecturing part-time at Athlone Institute of Technology and UCD Professional Academy.

AB

Dr. Amy de Buitléir joined Ericsson in 2011, where she works as a research scientist. Amy holds a PhD from Athlone Institute of Technology; as part of her PhD research she developed a species of artificial life agents with artificial intelligence that are capable of performing a variety of data mining tasks.

Dr. Enda Fallon is the Head of Department in Computer Science at Athlone Institute of technology (AIT), he also worked in as a system architect. In 2003 he founded AIT's Software Research Institute (SRI). Since 2003, Enda has been a principal investigator on over 60 collaborative industry/academic research projects valued at €6.2M. He has published over 70 peer reviewed articles in leading conferences and journals. His research interest focuses on service mediation and adaptation for heterogeneous networking environments.