A Programming-Based Approach to a Literature Review of Multi Agent Systems

Seminar of Computational Engineering, Department of Physics, University of Aveiro
Advisor: Professor Fernão Abreu

Rita Ferrolho - 88822
ritaferrolho@ua.pt

***Abstract*—Multi-Agent System (MAS) are intelligent systems that apply diverse Artificial Intelligence (AI) techniques from different subfields. These systems have been useful to industries, institutions, and research centers from a wide variety of domains. The main goal of this paper is to determine the steps taken to determine the most popular topics being discussed within the context of MAS, and review those topics. Some of these steps include collecting data through Web Scraping, and processing and analysing the collected data with Natural Language Processing (NLP), with an emphasis on Non-negative Matrix Factorization (NMF).**

***Index Terms*—Multi-Agent Systems (MAS), Web Scraping, Natural Language Processing (NLP), Non-negative Matrix Factorization (NMF).**

I. Introduction

In this section, the concept of Multi-Agent Systems (MAS) will be explained, as well as its applications.

A. Definition of Multi-Agent Systems

The theme that will be reviewed in this article is MAS, a subfield of Distributed Artificial Intelligence (DAI). This system is composed of multiple agents interacting between each other in a loosely coupled network, with the goal of solving problems that are beyond the individual capabilities or knowledge of each agent [1]. Therefore, MAS is different from a Single-Agent System (SAS), where each agent works individually, without recognizing nor interacting with other agents.

In the context of MAS, an individual agent is an entity which gathers data from its environment, extracts information from that environment through sensors and acts upon that information through actuators, depending on its goal. This is achieved by using varied Artificial Intelligence (AI) techniques, such as Machine Learning (ML), knowledge representation, reasoning, among others. A diagram that represents an agent can be observed at Figure 1.

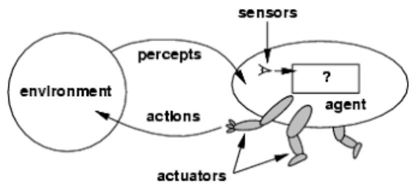

Fig. 1. An individual agent in the context of MAS [2].

These systems are capable of solving complex problems by recursively dividing them into sub-problems, until the sub-problems become simple enough to be solved. Then, once those atomic problems are solved, a solution to a complex problem is obtained by recursively assembling them back together. This is an algorithm design paradigm known as Divide and Conquer, which is represented in Figure 2. By applied this algorithm to MAS, a more efficient and intuitive solution to build intelligent systems is provided.

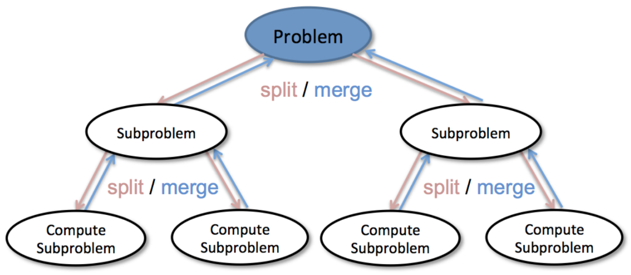

Fig. 2. Divide and Conquer algorithm [3].

B. Applications of Multi-Agent Systems

When it comes to applications, MAS can be applied to several different domains. According to a survey written in 2018 [4], some of these applications include:

- Computer networks - Perhaps one of the broadest applications, these structures are getting more complex with the increase of the number of connected devices to the Internet, resulting in the emerging field of Internet of Things (IoT). MAS can be very helpful in overcoming these complexities, from learning and detecting security threats to otimizing package delivery in routing protocols;
- Robotics - Cooperation and coordination between robots, as well as planning their trajectory, are some of the core challenges in this field, which can be solved through hardware and software-based agents. For example, MAS has been applied to robots in the context of soccer games;
- Modeling - Applying traditional equations to complex and dynamic systems can be costly and require significant processing. A solution known for this issue is Agent Based Modeling (ABS), a rule-based method that is low-cost and requires less processing. MAS, particularly ABS,

can be applied to smart grids in the power industry, smart cities, smart buildings, and smart homes.

II. Data Collection

In this section, the concept of Web Scraping will be explained, the failed attempts of gathering data will be discussed and the successful solution will be described.

A. Definition of Web Scraping

Web Scraping is a technique used to automatically extract data from websites, typically by requesting Hyper-Text Markup Language (HTML) files, parsing those files, and storing the data into a dataset. This technique has been useful in this work to search for articles and gather its related data.

B. Unsuccessful Attempts

Initially, it has been intended to apply Web Scraping to Google Scholar, a search engine used to find academic resources and scholarly literature. However, there were some issues in this approach, including:

- Websites with unpredictable structures - In the searches result of Google Scholar, the full abstract of each article did not appear. This problem can be solved by requesting the link to the page of each individual article and obtain its content, a task that was accomplished without much difficulty. However, the real issue was in parsing the HTML text, that is, obtaining particular data within the content. Two web pages of two different articles, obtained through the search results of Google Scholar, may have a very different HTML structure. In order to attempt to overcome this, the following implementation was considered: from the search results, store the incomplete abstract of an article, navigate to the web page of that article, and find the incomplete article within the HTML text. Then, find the markers which wrap the full abstract, and get the full abstract. Due to its extreme inefficiency, this solution was abandoned.
- Google Scholar detects and blocks bots - To prevent bots from scraping data or doing any other activity that may be suspicious from a security perspective, Google Scholar detects the speed at which an user browses the web. If the speed at which the user is browsing too high to assume it is human activity, then it is assumed that a bot is performing autonomous activity. As a consequence, the bot gets blocked and is unable to perform tasks temporarily. This is an inconvenience that is usually prevented by using proxies to route requests through different Internet Protocol (IP) addresses, setting different request headers, setting random intervals between requests, among a few other methods [5]. However, no balance between simplicity, speed and avoiding getting blocked was found, so these strategies were eventually dropped.

For these reasons, a different search engine was chosen, IEEE Xplorer. This search engine has greater consistency between different article websites, and neither tracks nor blocks bots. Building a web scraper for this search engine required less effort. At the same time, the developed bot was capable of several articles and abstracts in a shorter time, hence improving its speed. To summarize, by choosing a different search engine, the best of both conditions has been accomplished.

Since the web scraper has been developed with Python, a general-purpose programming language, the following modules have been explored, although unsuccessfully, both in Google Scholar and IEEE Xplorer:

- Selenium WebDriver - Selenium is an open-source tool that automates web browsers. Originally, this tool was developed to help software testers create automated tests that interact with interfaces of web applications. However, it can be applied to other contexts, including web scraping. This automation browser is useful to control and scrape dynamic websites, where the rendered content may also be rendered by JavaScript, instead of only HTML. In this work, the solution has been implemented not for scraping web pages of Google Scholar, but to gather data from IEEE Xplore, a similar search engine for academic work that has been published to the Institute of Electrical and Electronics Engineers (IEEE). However, having simulate user activity by automatically opening several web pages implies low efficiency, so this tool was not used in the final solution.
- Beautiful Soup - This is a python package that is helpful in parsing HTML documents. The module has been explored and applied with success, but due to its relative steep learning difficulty and ineffectiveness, this module was not used in the final implementation.

C. Successful Solution

The final developed Web Scraping solution required a minimalist but effective approach. Instead of using Google Scholar, where the websites of each query result have vastly different structures, a search engine with results that link to web pages with similar structures has been chosen, IEEE Xplore. Since this search engine does not track and block bots, many operations were no longer necessary, such as using proxies to change IP addresses regularly. Instead of using Selenium Web Driver and Beautiful Soup python modules, which require more effort to be implemented and takes a longer time to extract data, only the requests module has been imported. The usage of fewer modules and a reduced number of operations to collect data, and its relative speed in comparison with other explored alternatives, implies that a clean code has been achieved.

As in most projects, the developed code was not written from scratch. Instead, by starting with a basic web scraper that prints all the article titles from the query results of the first page [6], the code was adapted to gather more articles and more data from each article. This was achieved by wrapping the code into a loop from the first page to page 1000, collecting several data about each article, including the full abstract, and writing the data into a Comma Separated Value (CSV) dataset.

Details about the gathered data will be discussed in the next section.

III. Data Analysis and Processing

In this section, the structure of the obtained dataset will be described, the plots regarding the collected data will be presented, the NLP algorithms will be explained, and the results will be discussed.

A. Description of Dataset

The raw dataset contains more than 17800 rows and 10 columns. Each row corresponds to a different article and each column corresponds to the characteristics of an article, as described below:

- $page\_num$ - The page number of the query results at which the article appeared;
- art_num - Unique value that represents the id of the article;
- $type$ - Type of article, such as Conference Paper or Journal Article;
- $date$ - Date at which the article was published;
- $year$ - Year at which the article was published;
- num_cits - Number of articles who cited the article taken into consideration;
- num_dwnlds - Number of downloads of an article;
- $authors$ - All the authors of the article;
- $title$ - Title of the article;
- $abstract$ - Full abstract of the article.

The Figure 3 shows the printed output of a brief summary of the raw dataset.

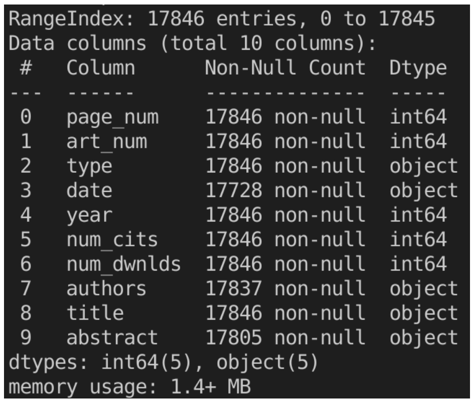
```
RangeIndex: 17846 entries, 0 to 17845
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   page_num    17846 non-null  int64
 1   art_num     17846 non-null  int64
 2   type        17846 non-null  object
 3   date        17728 non-null  object
 4   year        17846 non-null  int64
 5   num_cits    17846 non-null  int64
 6   num_dwnlds  17846 non-null  int64
 7   authors     17837 non-null  object
 8   title       17846 non-null  object
 9   abstract    17805 non-null  object
dtypes: int64(5), object(5)
memory usage: 1.4+ MB
```

Fig. 3. Summary of the raw dataset, which contains empty values.

A relevant information to notice from the previous summary is that there are null values in the columns $date$, $authors$ and $abstract$. The reason why there is missing data in the dataset is not because of a bot failure, but because that data is missing in the article website. For instance, the article "IEE Seminar on Autonomous Agents in Control" [7] does not show its abstract in its web page, as well as the names of the authors.

Since it is not ideal to process a column of abstracts with empty values, all the rows with empty value in the $abstract$ column were removed. As for the $date$ and $authors$ columns, there has been some initial indecision between removing the rows with empty values in these columns, or leaving them raw. Due to the vast amount of articles collected, and in order to avoid dealing with null values, it has been decided to remove all rows with empty values. After these steps were taken, the resulting summary of the new data frame is obtained as shown in Figure 4. The data visualization and analysis will be realized upon this data frame, which contains data from over 17600 articles. To be precise, 163 rows had null values, and so they have been removed.

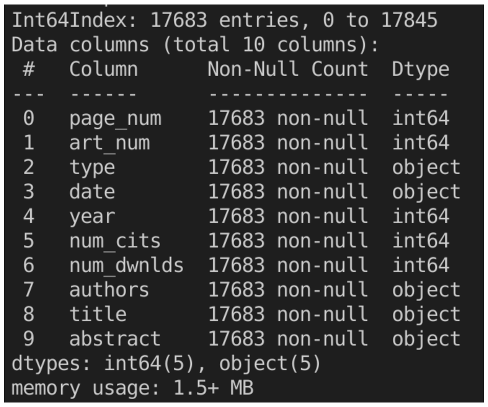
```
Int64Index: 17683 entries, 0 to 17845
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   page_num    17683 non-null  int64
 1   art_num     17683 non-null  int64
 2   type        17683 non-null  object
 3   date        17683 non-null  object
 4   year        17683 non-null  int64
 5   num_cits    17683 non-null  int64
 6   num_dwnlds  17683 non-null  int64
 7   authors     17683 non-null  object
 8   title       17683 non-null  object
 9   abstract    17683 non-null  object
dtypes: int64(5), object(5)
memory usage: 1.5+ MB
```

Fig. 4. Summary of the updated data frame, without empty values. This is the data frame that will be used for its analysis and processing.

B. Data Visualization

In every project that involves using algorithms to process and analyse a dataset, it is a good practise to get more familiar with the dataset, by plotting and visualizing its data, before applying the algorithms. One of the reasons for this procedure is that it helps to verify if the data has been properly gathered. This is done by checking if the plots are consistent with well-known facts, and whether they make sense or not.

For example, it is known that, in most fields of research, the number of Conference Papers is far greater than the number of Journal Articles. According to the bar plot displayed in Figure 5, that fact is confirmed. Therefore, it is likely that the values of the article types have been properly collected.

C. Natural Language Processing

This subsection is arguably one of the most crucial steps in this work, since it transforms data into information. In this particular case, words from each abstract will be captured so that they can be aggregated into clusters or groups, by applying NLP algorithms to those words. By taking this approach, insights about trends within MAS can be gathered. Some of the concepts related to NLP include stop words, lemmatization, stemming, and most importantly, Non-negative Matrix Factorization (NMF). All of these concepts will be explained in the next few paragraphs.

Stop words are a set of commonly used words that are perceived as irrelevant to the processing and analysis of patterns within text. This includes words such as "not", "and",

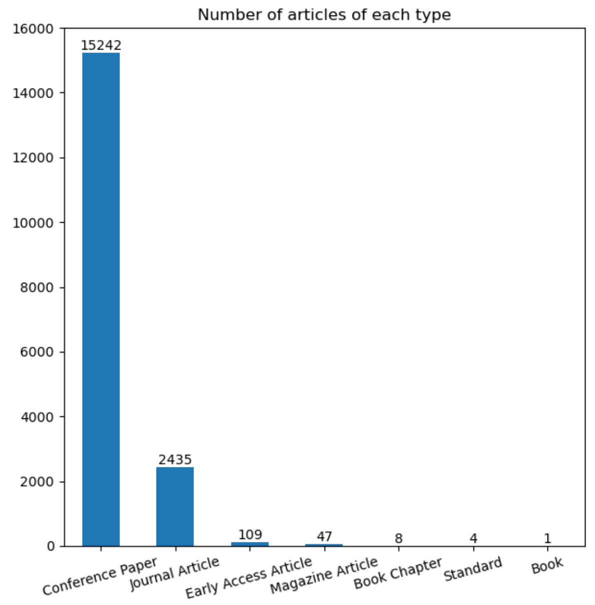

Fig. 5. Number of articles of each type. As expected, the number of Conference Papers is far greater than the number of Journal Articles.

”is”, ”are”, among others. In the context of this work, some specific words were introduced in one of the four implemented variations of NMF, which will be later discussed in this report.

Lemmatization consists of grouping together different inflected forms of a word, so that similar words can be considered to be the same word [8]. For example, the lemmatization of ”catches” is ”catch”, and the lemmatization of ”mice” is ”mouse”.

Stemming is the process of creating morphological variants of a root/base word [9]. For example, the word ”like” can be stemmed into ”likes”, ”liked”, ”liking”, among other related words. This concept is different from lemmatization, because lemmatization considers other factors beyond word reduction, such as the full vocabulary of a language to apply a morphological analysis to words, while stemming does not [10]. There are several variations of stemming, and after experimenting with different stemming approaches, the Snowball Stemmer has been chosen, due to its greater computational speed.

NMF is an useful method in Latent Semantic Analysis. The concept behind this methodology is that words with similar meanings appear in similar contexts, known as ”distributional hypothesis”. In this work, this method will be used to categorize the most common words (formally known as terms) taken from the abstracts into different clusters or groups.

According to its mathematical definition, NMF consists in the decomposition of the document-term matrix, which is associated with the visible variables (V) into two smaller matrices between the document-topic matrix, which represents the weights (W), and the topic-term matrix, which corresponds to the hidden variables (H). In other words, V matrix is the input, and the output is a factorization between the W matrix, which is a feature set, and the matrix H, which contains the coefficients. The matrixes W and H are populated with unnormalized probabilities. The Figure 6 illustrates this method.

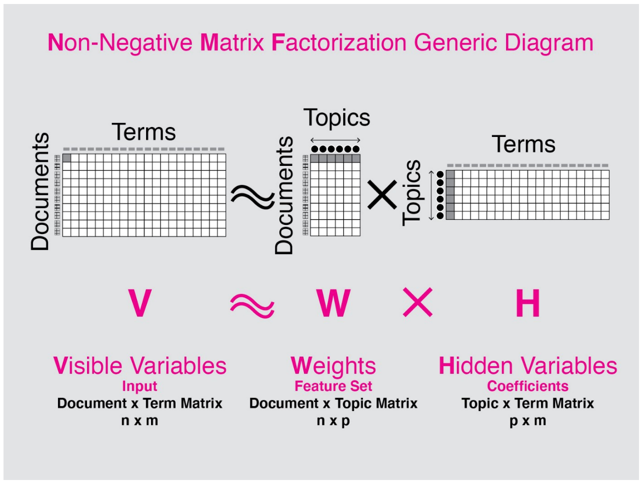

Fig. 6. Mathematical definition of the NMF method [11].

As for the implementation of NMF, some different variations of this method have been developed, but the common steps taken in these models were the following:

1) Removing stop words from the abstracts - For this step a python module has been used to remove terms that are typically considered to be stop words, such as ”and”, ”or”, and so on. However, in one of the developed models, some terms were manually included in the set. For example, it is not interesting to categorize the terms ”multi”, ”agent”, ”systems” and ”mas” into clusters, since it is already known that all articles are related to that theme, and so these terms were added to the set of stop words. Some other unnecessary words like ”paper”, ”throughout”, and ”finally” were also added to this set. Using these terms as stop words was only achieved in one of the four developed models, due to technical difficulties.

2) Lemmatization and Stemming - Lemmatization does not simplify some terms that are deemed to be similar, like for instance, ”playing” and ”play” are considered to be different terms. However, stemming tends to associate several terms to a particular one, so its context or meaning is sometimes lost. To solve this, a combination of both procedures has been made: after applying lemmatization to a term, that term is compared to its stemmed term. If the stemmed term is a substring of the most frequent lemmatized term, then the lemmatized term is returned; else, the stemmed term is returned. In spite of being a slow algorithm, its accuracy is increased.

3) NMF - In this step, a few different models have been used, including Latent Dirichlet Allocation (LDA), Frobenius norm and generalised Kullback-Leibler divergence [12]. In

The results of the execution of these models will be soon discussed.

D. Results

For all models, the chosen number of clusters is five and the number of terms per cluster is ten.

The results of the first implemented model, with extra stop words that were manually added, are as the following:

- Topic 0 - consensus, control, matrix, sufficient, condition, problem, protocol, lyapunov, result, linear;
- Topic 1 - based, user, intelligent, application, technology, network, software, environment, cooperative, management;
- Topic 2 - point, random, convergence, probably, bounded, illustrate, theorem, assumption, mean, given;
- Topic 3 - velocity, desired, composition, entire, evolution, rendezvous, supposed, smooth, group, move;
- Topic 4 - event, action, phase, multiple, identification, verified, nonlinear, characteristic, developed, computer.

Some trends can be spotted within this model alone. For example, the Topic 0 is related to the Lyapunov equations and the Topic 3 is related to the Rendezvous problem.

As for the other models, the results are plotted in Figure 7, Figure 8 and Figure 9. Due to formatting issues, only the two most relevant clusters will be shown, for each model.

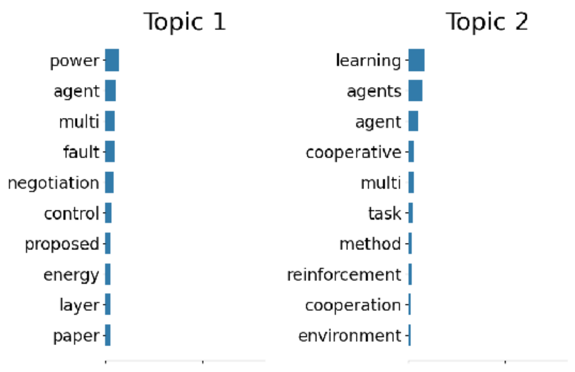

Fig. 7. Results of LDA model.

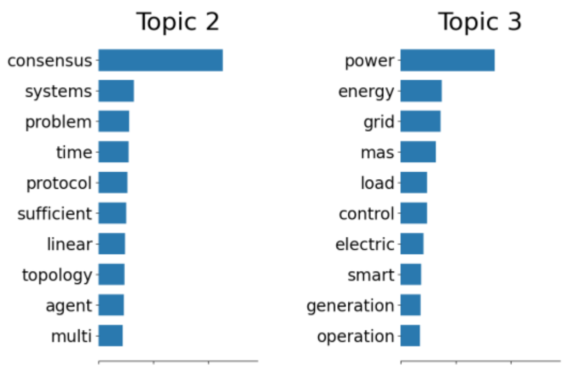

Fig. 8. Results of NMF model (Frobenius norm).

Some other trends can be detected in these plots, namely:

- Smart grids - Power and energy terms appear in Topic 1 on the LDA model results, and the terms smart, power, energy and grid appear in Topic 3 of Frobenius norm. This topic has been mentioned in the introduction of this report;
- Control - One of the most common detected terms in all articles;
- Agent Based Modeling - this topic was detected in the generalized Kullback-Leibler divergence model, a topic thta has been discussed in the introduction of this report.

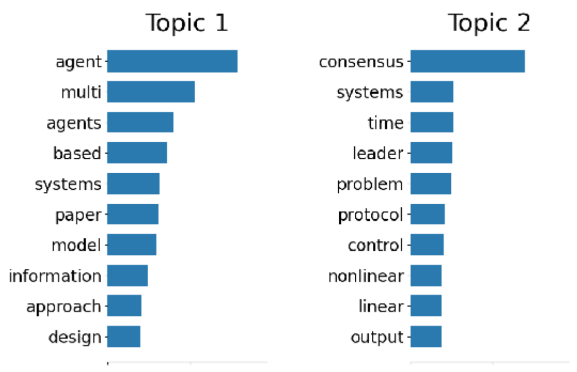

Fig. 9. Results of NMF model (generalized Kullback-Leibler divergence).

IV. Literature Review of trends within Multi Agent Systems

In this section, some of the spotted trends of MAS will be explained.

- Lyapunov equations that depend on communication graph topologies provide building blocks of Lyapunov functions, which play an important role in controller design and stability analysis of multi-agent systems [13]. Lyapunov stability theory analysis and based on the above argument, the equilibrium point of the multi-agent systems is asymptotically stable. Based on the Lyapunov stability theory, it is strictly proved that both transient and steady-state performance on consensus errors and full states can be guaranteed by introducing a novel asymmetric tangent barrier Lyapunov function.
- The Rendezvous problem of multi-agent systems originated from the nature behaviors of the social animals, the study of which is well motivated by the real applications such as mobile sensing network, intelligent transportation system and so on, see the survey paper [14].

V. Conclusion

Some trends spotted in previous surveys are common with those spotted in this report, such as smart grids and agent based modeling. However, other topics are emerging, such as Lyapunov equations and the Rendezvous problem.

Further work could have been done in this report. For instance, a more detailed review of the trends within MAS could have been made.

References

[1] (2006) Dr. Mevludin Glavic, “Agents and Multi-Agent Systems: A Short Introduction for Power Engineers”, Journals and Magazines, IEEE Acess, vol. 6, pp. 28573-28593, May 2006.

[2] (Spring 2008) Pinar Duygulu, “Chapter 2 - Intelligent Agents”, Accessed: 21 Fev. 2022 [Online]. Available: http://www.cs.bilkent.edu.tr/ duygulu/Courses/CS461/Notes/Agents.pdf

[3] (2022) Medium. Accessed: Feb. 19, 2022 [Online]. Available: https://medium.com/codex/divide-and-conquer-algorithm-f766640ef038

[4] Ali Dorri, Salil S. Kanhere, and Raja Jurdak, “Multi-Agente Systems: A Survey”, Journals and Magazines, IEEE Acess, vol. 6, pp. 28573-28593, April 2018.

[5] (2021) Medium. Accessed: Feb. 19, 2022 [Online]. Available: https://www.geeksforgeeks.org/web-scraping-without-getting-blocked/

[6] (2019) Stack Overflow. Accessed: Feb. 19, 2022 [Online]. Available: https://stackoverflow.com/questions/58463158/how-to-find-all-ieee-article-pages-to-scrape-using-beautifulsoup-just-all-artic

[7] (2005) Unknown author. ”IEE Seminar on Autonomous Agents in Control” Accessed: Feb. 20, 2022 [Online]. Available: https://ieeexplore.ieee.org/document/1499792

[8] (2021) Geeks for Geeks. Accessed: Feb. 20, 2022 [Online]. Available: https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

[9] (2021) Geeks for Geeks. ”Introduction to Stemming” Accessed: Feb. 20, 2022 [Online]. Available: https://www.geeksforgeeks.org/introduction-to-stemming/

[10] (2020) Towards Data Science. ”Stemming vs Lemmatization” Accessed: Feb. 20, 2022 [Online]. Available: https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221

[11] (2021) Towards Data Science. ”NMF — A visual explainer and Python Implementation” Accessed: Feb. 20, 2022 [Online]. Available: https://towardsdatascience.com/nmf-a-visual-explainer-and-python-implementation-7ecdd73491f8

[12] (2007-2022) Scikit Learn. ”Topic extraction with Non-negative Matrix Factorization and Latent Dirichlet Allocation” Accessed: Feb. 21, 2022 [Online]. Available: https://scikit-learn.org/stable/auto$_examples/applications/plot - topics - extraction - with - nmf - lda.html$

[13] (2015) Hongwei Zhang, Zhongkui Li, Zhihua Qu, and Frank L. Lewis, “On constructing Lyapunov functions for multi-agent systems”, Elsevie journal homepage, pp. 39-42, May 2015.

[14] (2018) Yi Dong, Youfeng Su, Yue Liu, and Shengyuan Xu, “An internal model approach for multi-agent rendezvous and connectivity preservation with nonlinear dynamics”, Elsevie journal homepage, pp. 300-307, March 2018.