INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Near Real Time Data Aggregation for NLP

TIAGO MIGUEL DA COSTA FERREIRA
Outubro de 2023

Near Real Time Data Aggregation for NLP

Tiago Miguel da Costa Ferreira

Aluno nº: 1200138

Dissertação para obtenção do Grau de

Mestre em Engenharia de Inteligência Artificial

Orientador: Doutor Luiz Felipe Rocha de Faria, Professor Coordenador do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Co-orientador: Doutora Maria Goreti Carvalho Marreiros, Professora Coordenadora com Agregação do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Júri:

Presidente:

Doutor António Constantino Lopes Martins, Professor Adjunto do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Vogais:

Doutor Luiz Felipe Rocha de Faria, Professor Coordenador do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Doutor Luís Manuel Silva Conceição, Investigador do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Porto, outubro 2023

Resumo

Com o aumento do uso das redes sociais, o número de opções de rede para usar e a variedade de funcionalidades que elas permitem leva à necessidade de os gestores desportivos prestarem uma atenção especial a estes meios. É seguindo este pensamento que surge o Projeto PLAYOFF e consequentemente esta tese.

Foi feito um levantamento da literatura existente de soluções que combinam Apache Kafka com modelos de machine learning e foi possível verificar que, apesar de soluções diferentes, já existem referencias nesses domínios.

É apresentada uma comparação entre Apache Kafka e RabbitMQ e as razões da escolha ter recaído para o Kafka. É apresentada de forma geral uma arquitetura de um projeto Kafka e, posteriormente, as diferentes abordagens pensadas e desenvolvidas no âmbito da dissertação, assim como o formato das mensagens trocadas usando este sistema.

Uma serie de testes e seus resultados são descritos, de modo a comprovar a sua escolha e utilização. Nestes testes diferentes abordagem de execução paralela (threads e processos) são apresentadas, assim como a forma de obter dados das APIs das redes sociais também possui diferentes abordagens.

As alterações que foram realizadas aos modelos originais são descritas e explicadas as razões para essas mudanças e de que forma se enquadram na ferramenta desenvolvida.

Foi realizado um teste global e final, designado por “Teste Piloto”, onde em ambiente real, com um evento real foram testados todos os componentes deste projeto, incluindo os sistemas externos desenvolvidos pela MOG Technologies e os componentes desenvolvidos no âmbito desta dissertação.

Por fim, é possível comprovar as soluções apresentadas e opções finais escolhidas para o projeto, através dos resultados obtidos nos diferentes testes. É ainda proposto trabalho futuro de continuação do desenvolvido.

Palavras-chave: Processamento de Linguagem Natural, Análise de Sentimentos, Análise de Tópicos, Apache Kafka, Hospedagem de Modelos de Inteligência Artificial, Comunicação em Tempo Real

Abstract

With the increasing use of social networks, the number of network options to use and the variety of functionalities that they allow leads to the need for sports managers to pay special attention to these media. It is following this thought that the PLAYOFF Project emerges and consequently this thesis.

A search of the existing literature on solutions that combine Apache Kafka with machine learning models was carried out and it was possible to verify that, despite different solutions, there are already references in these domains.

A comparison between Apache Kafka and RabbitMQ and the reasons for choosing Kafka are presented. A general architecture of a Kafka project is presented, as well as the different approaches thought and developed within the scope of the dissertation, as well as the format of the messages exchanged using this system.

A series of tests and their results are described, in order to prove their choice and use. In these tests different parallel execution approaches (threads and processes) are presented, as well as the way of obtaining data from the APIs of social networks also has different approaches.

The changes that were made to the original models are described and explained the reasons for these changes and how they fit into the developed tool.

A final and global test was carried out, called “Pilot Test”, where in a real environment, with a real event, all the components of this project were tested, including the external systems developed by MOG Technologies and the components developed within the scope of this dissertation.

Finally, it is possible to verify the solutions presented and final options chosen for the project, through the results obtained in the different tests. It is also proposed future work of continuation of the developed.

Keywords: Natural Language Processing, Sentiment Analysis, Topic Analysis, Apache Kafka, Hosting of Artificial Intelligence Models, Real-Time Communication

Acknowledgement

I would like to express my sincere gratitude to all the people who have supported me throughout my master thesis project. First and foremost, I would like to thank my supervisors and Professors, Luiz Faria and Goreti Marreiros, for their invaluable guidance, feedback, and encouragement. They have been very patient and helpful in every step of the process.

I would also like to thank GECAD for providing me with the opportunity to work on PLAYOFF Project which was both challenging and rewarding. I am grateful for the access to the data, resources, and facilities that enabled me to conduct this research.

I am indebted to my friends, both at ISEP and elsewhere, for their constant support, advice, and friendship. They have made my master journey more enjoyable and memorable. I would like to mention João Oliveira in particular, for being there for me whenever I needed them.

Finally, I would like to dedicate this thesis to my family, especially my mother, father and sister, who have always been my source of inspiration and motivation. I cannot thank them enough for their unconditional love, care, and sacrifice. They have always believed in me and encouraged me to pursue my dreams. This thesis would not have been possible without them.

Index

List of Figures

List of Tables

List of Code

List of Acronyms

CNN	Convolutional Neural Network
DL	Deep Learning
AI	Artificial Intelligence
RF	Random Forest
RGPD	Regulamento Geral sobre a Proteção de Dados
SA	Sentiment Analysis
NLP	Natural Language Processing
SVM	Support Vector Machines
I/O	Input/Output
IPC	Inter-Process Communication
PRAW	Python Reddit API Wrapper
UEFA	Union of European Football Associations
VADER	Valence Aware Dictionary for Sentiment Reasoning
ETL	Extract Transform Load

1 Introduction

This document details the work carried out in building the hosting infrastructure for the "Natural Language Processing" component of the PLAYOFF project. It also details subsequent work on this component, namely a statistical summary model of the results obtained. Also details some improvements to the NLP models.

On this chapter, the context of the project, the problems that will be trying to solve and the main goals and expected results of the work will be provided and contextualized.

1.1 Context

This dissertation is part of the Playoff Project. The work to be carried out arose, first of all, from the need to place the sentiment analysis and topic analysis models developed in an earlier phase and by another author, in a structure that would allow their operation and communication with MOG. After these were working correctly, there was also the need to perform a statistical summarization of the analysis results of both machine learning models.

Lastly, it was necessary to investigate improvements to be implemented in the models, particularly in the topic analysis model, where its author mentioned some of those improvements to be made (Ferraz Barbosa Soares de Albergaria, 2022).

1.1.1 Playoff Project

The PLAYOFF project – "Personalized LAYered multi-source content – Optimized with data Fusion topologies for sports Fans", is a project developed in partnership between Instituto Superior de Engenharia do Porto (ISEP)[1], more specifically by its research group in the area of

[1] ISEP Website - https://www.isep.ipp.pt

artificial intelligence, GECAD[2] – "Research Group on Intelligent Engineering and Computing for Advance Innovation and Development" – and the company MOG Technologies[3].

The main objective of this project is the creation of a reactive multimodal media transmission environment, aimed at television stations and clubs. This system has several important features:

- **Cloud ecosytem** with a personalized experience for each user (television stations and clubs) based on the fusion of different sources of live multimedia sports content.
- **Integrated environment** able to provide real-time data analytics services using audience analysis and content consumption patterns, including data taken from social networks.
- **Multilayer toolkit** for effective management and content enrichment and multimodal event summarization.

1.1.1.1 Description

The project has several stages and phases, being the author responsible for the continuation of the previous work executed in GECAD. This work was carried out by Miguel Albergaria for his master's thesis (Ferraz Barbosa Soares de Albergaria, 2022) and consisted of the NLP part of the project, where he developed a sentiment analysis model and another topic analysis model.

The work in the present document consists, firstly, in the elaboration of a communication structure based on Kafka, where information sent by the MOG is received in the ISEP servers. This information indicates which channel and corresponding social network to be analysed. There is also the possibility of analysing comments from an internal chat, which we will refer to as the Chat Platform from now on in this document, placed on the MOG platform. Subsequently, the inference of the analysis of sentiments and topics is carried out. This structure will be explained in greater detail in the sub-chapter X of this document.

Later, the main part of the project was developed by the author and consisted of the elaboration of statistical models for summarizing the results of the inference models for the analysis of sentiments and analysis of topics.

Finally, an inspection of the topic analysis model was carried out (where the author's notes were also considered) to determine improvements to be made in it and carry them out.

The developed modules and the Kafka structure use the python[4] programming language.

1.1.1.2 Previous Work

The project has several stages and phases, being the author responsible for the continuation of the previous work executed in GECAD. This work was carried out by Miguel Albergaria for his

[2] GECAD Website - https://www.gecad.isep.ipp.pt
[3] MOG Technologies Website - https://www.mog-technologies.com
[4] Python Website - https://www.python.org/

master's thesis (Ferraz Barbosa Soares de Albergaria, 2022) and consisted of the NLP part of the project, where he developed a sentiment analysis model and another topic analysis model.

The document (Ferraz Barbosa Soares de Albergaria, 2022) presents the development of a near real-time sentiment analysis tool and a near real-time topic analysis tool for the analysis of social media content related to sports events that was published during the transmission of the respective events, thus enabling, in near real-time, the understanding of the sentiment of the viewers and the topics being discussed through each event.

To create both tools, a study was conducted on social media platforms with the greatest potential to be used as the primary data source. Twitter was chosen due to its availability of high-quality text content about most sports events and its ability to stream data rather than just provide it via GET requests.

To develop a new tool, a study was conducted on the literature of both sentiment and topic analysis fields. The best dataset found had sentiment labels for all its 6.3 million data points but no topic labels. As a result, the DistilBERT, DistilRoBERTa, and Knowledge embedding of these pre-trained supervised models were found to be the most promising methods for sentiment analysis, while the JoSH and WeSHClass weakly supervised models were found to be the most promising methods for topic analysis.

Before testing and subsequently selecting the best processing technique for each task, the best set of pre-processing steps was tested due to the models outputs being highly dependent on the quality of the data provided to them. This allowed for the selection of the most appropriate pre-processing for each task and led to the conclusion that the topic analysis task benefits more from less complex data than the sentiment analysis task. Additionally, it was concluded through experimentation with several combinations of pre-processing steps that the simple replacement of emojis and emoticons for their respective tokens has very minimal benefits for sentiment analysis.

After selecting the pre-processing steps, the sentiment analysis models were tested, with the knowledge embedded DistilRoBERTa model achieving the best results. As a result, an accuracy of 94.7% was achieved by training this model on the unsampled dataset, which was simplified with the best combination of sentiment analysis pre-processing steps. The average combined pre-processing and processing time was 0.1769 seconds per 100 instances.

Regarding topic analysis, due to the lack of an inference function on the JoSH model, the WeSHClass model was chosen. This model was pre-trained for five epochs on 500 pseudo documents generated using Skip-gram and an LSTM model. Trained on the dataset simplified with the best combination of topic analysis pre-processing steps, this model achieved a weighted F1 score of 0.61 on the classification of 11 topics, with an average combined pre-processing and processing time of 0.770167 seconds per 100 instances. Although this F1 score is not very high, when examined through its confusion matrix, it was concluded that it had to do with similar topics overlapping, such as save and penalty or foul, red card, and yellow card,

thus being misleading and making it seem that the topic analysis performance is worse than it really is.

It was concluded through an initial unsuccessful approach that both the more traditional unsupervised statistical techniques such as LDA, NMF, and LSA and the coherence score evaluation metric is unsuitable for the intended topic analysis tool. As a result, the dataset was partially annotated regarding the topics to enable the use of weakly-supervised hierarchical topic models coupled with the F1 score.

1.1.1.3 Emotions and Topics from the NLP Models

The machine learning models developed in (Ferraz Barbosa Soares de Albergaria, 2022), have the following emotions and topics in the results:

Emotions

- Negative
- Neutral
- Positive

Topics

- Save
- Foul
- Free_kick
- Goal
- Kickoff
- Offside
- Penalty
- Red_card
- Yellow_card
- Back_pass
- Substitution

1.2 Problem description

Given the work done previously and the objectives of the project, an infrastructure was needed to host the machine learning models and to allow them to run under the best conditions and always in communication with MOG, in order to receive and process the requests.

Currently, there are already platforms to deploy machine learning models, namely from technological giants such as Google, Microsoft, or Amazon (Fergus & Chalmers, 2022). But the goal was to deploy both models to a server in ISEP. In this way, all models would be on private servers and on one of the project's stakeholders. Which helps to mitigate the discrepancy issues between development and deployment on servers (Chen, 2021; Heymann et al., 2023). As this

approach was decided upon at the planning stage, the entire project was designed to work this way, which did not cause any problems (Heymann et al., 2022; Weiner, 2021). In addition to hosting the models, it was also necessary to create a communication infrastructure between the MOG platform, where users indicate which games and social networks will be analysed, and the ISEP server where the analysis models will be hosted.

After obtaining the results of both models, it was found that they have no context or any kind of statistical analysis that would summarize the analysis. Thus, a model was also though out and developed to perform this analysis on the results of the models.

As describe by the author of the models in (Ferraz Barbosa Soares de Albergaria, 2022), they have improvements that can be made, and one was thought out for the topic analysis model.

1.3 Main goals

Within what is requested in the project as a whole and after what is described in the problem description, the objectives are the completion of these three parts/phases of the project.

Create a hosting structure for the machine learning models capable of running them without problems. In addition to their execution, it must ensure that communication between the hosting servers and the MOG servers is active so that execution can be carried out as intended.

While in execution is taking place and after obtaining results, this is necessary to present a summarization of these results from each analysis and for this a statistical model will be developed. In this way, the results will be easily understandable to everyone and will have valid and comparative significance, something that responds to one of the global problems and objectives of the PLAYOFF project.

That being said, the results of the sentiment analysis and especially the topic analysis models can be improved, so improvements for this model will also be studied, presented and carried out. This will also improve the statistical summarization of the analysis.

1.4 Expected results

Once the development of this work has been completed, it is expected that all the objectives will be achieved, that is, the structure for housing the sentiment analysis and feeling analysis models will be functional, it is also expected that the communication system between MOG and ISEP is operational using a communication framework. Finally, it is expected that context will be given to the results of the analysis.
The project will be developed in such a way that it is possible to carry out future work on what was developed in it and this possible continuation will be facilitated, indicating, for example, where improvements can be made and what challenges and limitations were encountered.

1.5 Contributions

The development of the goal of this project mainly contribute to the continuation of the development of the "PLAYOFF" project, but also contribute to their respective fields by:

- Providing an analysis of existing communication tools and the characteristics of each one.
- Creating a model structure that host two machine learning models and run them in a server environment.
- Providing a literature review of statistical analysis of the results of sentiment and topic analysis.
- Creating a model of summarization of results of sentiment and topic analysis.
- Providing improvements to the topic analysis model, specifically dedicated to short term texts.

1.6 Dissertation Structure

In this subsection the structure of this document will be described. This dissertation has seven main chapters: Introduction, State of the Art, Methods and Materials, Communication Infrastructure, Changes to NLP models, Results of Pilot Test and Conclusion.

On the first chapter, which is this one, the Introduction, the context, the main goals, and the expected results are described. As well as the problem that the developed project intends to solve and who contributed for this problem and project to be developed. Also, the Playoff Project is described in here.

After the Introduction, in the second chapter there will be a study of social networks and describe the chosen ones and the reasons for this choice. Their operation will also be discussed, as well as obtaining data from them. And describe some literature found that is related to this project.

The third chapter, called Methods and Materials, two communication frameworks are compared and the reasons for choosing Kafka are enumerated. In the end, there's also a sub-chapter about ethical problems with AI.

On the fourth chapter is the main one. Is here that the configurations used are showed, the different types of Kafka Messages, their parameters and the reasons why was opted to do that way. All the approaches of the architecture structure are described and the differences between them. Lastly, the way go get data from the social networks are explain and the alternatives to improves the time of execution doing so is presented.

The fifth chapter is the one, where the changes needed to be made to the original models are presented and the reasons for those changes and comparison of results of the original vs the changes made.

The chapter six, called Results of the Pilot Test, presents the last test of this whole project, we called “Pilot Test”. It was done in real time and is described in this chapter.

The last main chapter is the Conclusion. It is the seventh chapter, and every result is discussed, and meaning is given to those results. Some of the future work, will be described in this chapter as well.

2 State of the Art

This chapter addresses the state of the art of the subjects matters used and described for the development of the project detailed in this document. Those subjects are social networks, social media in sports, text mining, more specific sentiment analysis and topic analysis. Communication frameworks and architectures used to feed machine learning models, more specific NLP models, in servers.

2.1 Social networks and social media in sports

Social network analysis originated during the first half of the twentieth century in the disciplines of psychology, sociology, social psychology, and anthropology (Knoke, 2014). Core concepts and principles of social structural relations were developed by a handful of scholars (Knoke, 2014).

Many people like to link the history of social media to the growth in communications technology that has been occurring since the end of the 19th century (Jones, 2015). A common starting point is Samuel Morse's first telegraph, which he sent in 1844 between Washington, D.C. and Baltimore (Jones, 2015). The earliest forms of social media appeared almost as soon as technology could support them (*Social Media | Definition, History, Examples, & Facts | Britannica*, n.d.). E-mail and chat programs debuted in the early 1970s, but persistent communities did not surface until the creation of the discussion group network USENET in 1979 (Ray, n.d.; *Social Media | Definition, History, Examples, & Facts | Britannica*, n.d.).

The reason Six Degrees is the first of the social networks is because it allowed people to sign up with their email address, make individual profiles, and add friends to their personal network (Jones, 2015). It was officially launched in 1997, and it lasted until about 2001 (Jones, 2015). Its number of users peaked at around 3.5 million (Jones, 2015).

Social media is an important part of modern life and has an interesting history. The evolution of social media began with the emergence of sites like Friendster and Myspace, which allowed family members, friends, and acquaintances to connect online (*Social Media | Definition,*

History, Examples, & Facts | Britannica, n.d.). However, these two sites were eventually supplanted by Facebook, which became one of the world's most popular social media sites with billions of users worldwide (*Social Media | Definition, History, Examples, & Facts | Britannica*, n.d.). Since the beginning of the century multiple social media platforms have been appearing and disappearing (Stavros et al., 2014) and each one had different functionalities and different grown rate (Chawinga, 2017).

Social media has been studied by scientists in various disciplines including computer science, psychology, and sociology (Bik & Goldstein, 2013). These studies have shown that social media can have a significant impact on people's lives and on society as a whole (Dizikes, 2020). For example, social media can be used to disseminate important information quickly and to connect people with similar interests (Bik & Goldstein, 2013). However, social media can also be used to spread misinformation and hate (Dizikes, 2020).

In summary, social media has an interesting history and is an important part of modern life. However, it is important to remember that it has both benefits and drawbacks.

Some of those benefits and drawbacks are specified in (Drahošová & Balco, 2017). It discusses the benefits and drawbacks of using social media. The research was conducted in European Union countries and found that the biggest advantage associated with social media use is information exchange and communication, followed by data sharing. However, the research also found that excessive use of social media can lead to mental health problems such as anxiety and depression (Drahošová & Balco, 2017).

Experts in the field of sports science have been researching the use and potential of social media in sports. Many studies have focused on social media platforms such as Facebook and Twitter and have demonstrated their significant potential and importance for the sports industry.

In (López-Carril et al., 2020) was possible to conclude that in recent years the number of studies of social media in sports has grown a lot. With the applications in various themes inside de sports industry. This show that is possible to find important information for this field in this type of platforms.

Social media has become an integral part of our lives, both personally and professionally. Approximately one third of the Earth's population was monthly active on social media in the year of 2021 (Dixon, 2023). Thanks to this rise of social media, is possible to create a lot more content of an event (Billings et al., 2018; Naraine et al., 2019). The way that some organizations communicate with their fans and peers also change because of social media (Li et al., 2019; Naraine et al., 2019)

In a study (Filo et al., 2014), seventy journal articles were analysed on the impact of social media in the field of sport management from a service-dominant logic perspective, with an emphasis on relationship marketing. The review social media three categories of social media research: strategic, operational, and user focused. The review shows that social media research in sport management aligns with service-dominant logic and illustrates the role of social media in cultivating relationships among and between brands and individuals. Interaction and

engagement play a crucial role in cultivating these relationships. The text also discusses opportunities for future research as well as suggestions for theoretical approaches, research design, and context.

For this project, the social networks that were taken into consideration before choosing the final two were Facebook[5], YouTube[6], Instagram[7], Twitter[8], Reddit[9], Tumblr[10], Twitch[11], and Discord[12] (Ferraz Barbosa Soares de Albergaria, 2022). Initially, the social media platforms chosen were Twitter and Reddit, but the final version of the project uses YouTube and Reddit.

2.1.1 YouTube

YouTube is a platform for sharing videos that allows users to upload, watch and share content. It was founded in February 2005 by Steve Chen, Chad Hurley and Jawed Karim. Currently is owned by Google and has over 2 billion active users (Belle Wong, 2023).

The YouTube API provides developers with the ability to access data from YouTube, which can be used to create a range of applications. The API provides access to a wealth of data, including video metadata, video statistics, and user data.

One of the characteristics that is important in YouTube, is the ability to allow livestreaming. It is in this context that YouTube comes into play. Comments made by users will be analysed in near real time.

2.1.2 Reddit

Reddit was founded in June 2005 by Steve Huffman and Alexis Ohanian. Reddit is a platform where users can create and manage their own communities, called subreddits. Subreddits can be about anything, from hobbies to interests to current events. Users can also submit content to subreddits, such as links, text posts, images, and videos. This content is then voted on by other users, and the most popular content rises to the top. The subreddit can also have inside them posts, and those can be about live events, like sports matches. Every comment or response in a specific event will be analysed.

[5] Facebook Website - https://www.facebook.com/
[6] YouTube Website - https://www.youtube.com/
[7] Instagram Website - https://www.instagram.com/
[8] Twitter Website - https://twitter.com/
[9] Reddit Website - https://www.reddit.com/
[10] Tumblr Website - https://www.tumblr.com/
[11] Twitch Website - https://www.twitch.tv/
[12] Discord Website - https://discord.com/

Reddit offers an API[13] that allows developers to access and interact with Reddit data. The API is a set of tools and instructions that developers can use to build applications that connect to Reddit.

2.2 Text Mining

Text mining is a process of extracting high-quality information from text, including discovering new information in it (*What Is Text Mining? | IBM*, n.d.). Nowadays it is a computer-based process and automatically extract the information, for example it allows researchers to extract valuable knowledge from large volumes of data (Isayev, 2019; *Text Mining Scientific Articles: Why You Need the Full Picture | CCC*, n.d.; Westergaard et al., 2018). Text mining can help write research papers or even master's thesis, like this one (Atanassova et al., 2019), but have many more applications. There are multiple text mining techniques such as Naïve Bayes, SVM or other deep learning algorithms is possible to get "hidden" information from text (*What Is Text Mining? | IBM*, n.d.) and define a structure to that data and give a context to the output. Some of the text mining tasks are information extraction (Hong et al., 2021; Nasar et al., 2018), text categorization (Dhar et al., 2021; Lin et al., 2014; Taha & Yousif, 2023), text clustering (Lin et al., 2014; Mishra et al., 2022), document summarization (Cohan & Goharian, 2018), sentiment analysis (Nandwani & Verma, 2021) or topic analysis (Kherwa & Bansal, 2019).

2.2.1 Sentiment Analysis

Sentiment analysis, a subfield of NLP, is the process of identifying and extracting subjective information in source material, such as opinions, attitudes, and emotions towards something (or entity). An entity is a person, event, or topic that can be the subject of sentiment analysis (Medhat et al., 2014). A range of approaches, strategies, and instruments are available for identifying and obtaining subjective data, such as views and sentiments, from language (Mäntylä et al., 2018; Nandwani & Verma, 2021). The process involves identifying the contextual polarity of the text and ascertaining if the given text is positive, negative, or neutral (Devika et al., 2016; Nandwani & Verma, 2021). It combines machine learning and natural language processing to achieve this (*What Is Sentiment Analysis? A Complete Guide for Beginners*, n.d.). Sentiment analysis is the contextual mining of text that identifies and extracts subjective information in source material, helping a business understand the social sentiment of its brand, product, or service while monitoring online conversations (*Sentiment Analysis: Concept, Analysis and Applications | by Shashank Gupta | Towards Data Science*, n.d.). Sentiment analysis is widely used in various industries for purposes such as monitoring social media, analysing customer support tickets, managing brand reputation, understanding the voice of the customer and employee, evaluating products, conducting market and competitive research (*8 Applications of Sentiment Analysis*, n.d.; *Applications of Sentiment Analysis - DataScienceCentral.Com*, n.d.; *Top 10 Applications of Sentiment Analysis in Business*, n.d.).

[13] Reddit API Overview - https://www.reddit.com/dev/api/

Sentiment analysis techniques can be broadly classified into three categories (Indhraom Prabha & Umarani Srikanth, 2019):

- Lexicon-based techniques
- Machine learning-based techniques
- Hybrid techniques

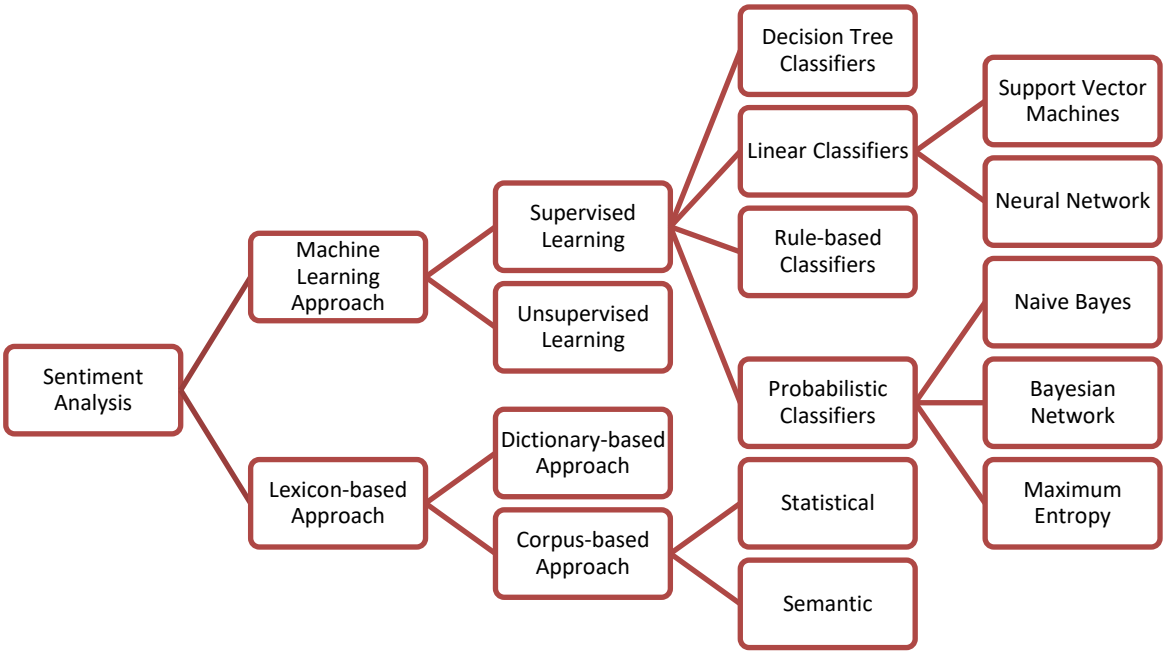

Figure 1 - Sentiment analysis most used techniques (Medhat et al., 2014)

exicon-based approaches rely on a pre-determined set of words that are linked to either positive or negative emotions (Medhat et al., 2014). Machine learning-based approaches employ algorithms like Neural Networks, Support Vector Machines, Naive Bayes Classifier, Maximum Entropy, and Bayesian Network (Medhat et al., 2014). Hybrid techniques combine both lexicon-based and machine learning-based techniques.

Sentiment analysis in social media is a popular research topic that has been explored by many researchers (Drus & Khalid, 2019) and can be used to understand public opinion on a wide range of topics (Nandwani & Verma, 2021). Social media platforms hold vast quantities of unprocessed data that users have shared in the form of text, videos, images, and audio. Sentiment analysis helps businesses understand how their brand is perceived by their customers and what they can do to improve their brand image or in this case the sport club (Drus & Khalid, 2019).

Sentiment analysis can be employed to assess the success of social media campaigns (Poecze et al., 2018). It can also be utilized to gauge the attitude of followers towards a brand (Poecze et al., 2018). When user-generated activity is relatively low, sentiment analysis can uncover underlying negativity among followers (Poecze et al., 2018).

In sports, sentiment analysis is a rapidly expanding field of study within computer science with applications in several domains (Wunderlich & Memmert, 2020). The research examines the potential of lexicon-based sentiment analysis tools in processing football-related text data on the Twitter microblogging platform. The results demonstrates that sentiment analysis can be an effective and useful tool for sports-related content.

2.2.2 Topic Analysis

Topic analysis is a machine learning method that categorizes and comprehends vast amounts of text data by assigning "tags" or categories based on the topic or theme of each text (Kherwa & Bansal, 2019; Vayansky & Kumar, 2020) . It is commonly used in NLP to automatically derive meaning from text by identifying recurring themes or topics. Topic analysis can be approached in two ways: Topic modeling and Topic classification (Kherwa & Bansal, 2019).

Topic modeling is employed to uncover the primary topics in a collection of texts, whereas topic classification is utilized to automatically categorize texts by topic (Alghamdi & Alfalqi, 2015; Kherwa & Bansal, 2019; Vayansky & Kumar, 2020). The approach you choose will depend on the problem you are trying to solve (Alghamdi & Alfalqi, 2015; Kherwa & Bansal, 2019; Vayansky & Kumar, 2020).

Topic analysis can be applied at different levels of scope, such as document-level, sentence-level, or word-level (Kherwa & Bansal, 2019; *Topic Analysis: A Complete Guide*, n.d.). Here are some applications of topic analysis (*Topic Modeling: Algorithms, Techniques, and Application - DataScienceCentral.Com*, n.d.):

- Classification, categorization, and summarization of documents.
- Analysis of social networks pertaining to the sentiments of users.
- Understanding large document collections by finding general themes present in the collection.

Topic analysis is most effective when applied to longer texts or documents, such as in scientific articles (O'Mara-Eves et al., 2015), but traditional models struggle to determine the themes and topics in social media posts or comments (Yan et al., 2013). These models tend to suffer from sparse data due to the small size of the text, resulting in less frequent word co-occurrence in each document, making it challenging for topic modeling to discern the meaning of ambiguous words in each context (Santos et al., 2022; Yan et al., 2013), this happens because of the nuances and complexities of human language (Westergaard et al., 2018).

2.3 Related Work

In recent years, projects have emerged that use Kafka and social networks to feed machine learning models.

One of the recent developments in the field of social media analysis is the project "TwitterAnalysis: Social Media Analysis using Apache Spark and Kafka" (Sengupta, 2019). The project uses Python as the main programming language and leverages the power of Apache Spark[14] and Apache Kafka to handle large-scale streaming data. Also uses the VADER (Hutto & Gilbert, 2014), to perform sentiment analysis. A Kafka producer is responsible for obtaining the data from Twitter, via Streaming API, provided by the social network. These tweets are sent to a Kafka consumer that will receive the messages with the comments and do pre-processing to the tweets and will feed the VADER tool. The results of this analysis are stored in the SparkSQL database. This project demonstrate how social media data can be used to gain insights into public opinions and feeling towards various topics. It also shows how tools like Apache Spark and Apache Kafka can facilitate real-time data processing and analysis, especially when combined with python which create a versatile application and can be combined with different libraries and frameworks.

Another similar solution is "Spark-Twitter-Streaming: A Real-Time Streaming ETL Pipeline for Twitter Data" (Ajith, 2020). This is a real-time streaming ETL pipeline that streams and performs sentiment analysis on Twitter data using Apache Kafka, Apache Spark and Delta Lake. The pipeline consists of two main components: a tweet stream producer and a tweet stream consumer. The pipeline consists of two main components: a tweet stream producer and a tweet stream consumer. The tweet stream producer uses Tweepy[15], a python-based library for accessing the Twitter API, to stream tweets and publishes them to a Kafka topic. A Kafka producer fetches tweets by topics and sends the obtained data to Kafka topics. Through a Kafka consumer and Apache Spark, the data are put in data frame format. These tweets undergo pre-processing and are placed in a machine learning model to perform sentiment analysis.

The article (McDonald, 2019) shows how to use Apache APIs to build a streaming machine learning pipeline for sentiment analysis. The authors use Amazon product reviews as streaming data and train a logistic regression model on them using Spark. The model can classify the reviews as positive or negative with high accuracy. The authors then use the model with Spark structured streaming to process new reviews from Kafka and enrich them with sentiment scores and metadata. The authors store the enriched reviews in MapR Database and query them with Spark SQL and Drill.

[14] https://spark.apache.org/
[15] https://www.tweepy.org/

3 Methods and Materials

In this chapter, the communication frameworks that were under analysis as a hypothesis for use, will be described. A comparison will be made between them. The chosen framework and its advantages and characteristics will be detailed. Ethical problems related to artificial intelligence will also be described.

3.1 Communication Frameworks

The two alternatives for the project were Apache Kafka[16] and RabbitMQ[17], are very popular open-source tools. These are tools that use the publish/subscribe method. This is a distributed system, where messages are sent by publishers (or producers) and subscribers (or consumers) will receive these messages. In this type of systems, communication is not individualized, that is, publishers produce the message and any subscriber who is connected to the corresponding topic/content will consume the message and has the great advantage of being very easily scalable and allowing a more dynamic network topology. These tools are used for real-time communication but differ from each other in several aspects.

Apache Kafka (*Apache Kafka*, n.d.) was created by LinkedIn to centralize the platform's real-time events (Goodhope et al., 2012). It has a large capacity and low latency in real-time data processing (Goodhope et al., 2012). Its focus is message streaming, and the result is a scalable system of transaction logs (Wang et al., 2015a).

RabbitMQ is a software that implements the AMQP protocol. This protocol was developed to address the existing needs in the handling of asynchronous messages (*Launch of RabbitMQ Open Source Enterprise Messaging*, n.d.). The final product has great performance, security and

[16] Apache Kafka website - https://kafka.apache.org/
[17] RabbitMQ website - https://www.rabbitmq.com/

allows for scalability (*Launch of RabbitMQ Open Source Enterprise Messaging*, n.d.). Besides this protocol, RabbitMQ also implements the STOMP protocol and MQTT.

A study conducted by (Dobbelaere & Esmaili, 2017) indicates the characteristics of each alternative. In terms of latency, the results are very similar, and both have low latency, which is intended. But there are some differences in terms of performance, RabbitMQ's queues are only faster when they are empty and to be able to process millions of data would require more resources, while Kafka processes millions of data per second easily.

The tools also differ in the way information is passed from the producer to the consumer. Kafka has a mechanism where the producer is dumb and the consumers smart, that is, the producer puts the message in a certain topic and does not know if any consumer will receive the message. Here it is the consumer who manages the topics to which he subscribes, and messages received. In RabbitMQ it's the opposite, the producer sends the message to the consumer and waits for an acknowledgment, to have the confirmation that the consumer has received it and the consumers only receive the messages sent to them by the producer.

To promote scalability and redundancy Kafka relies on partitions. A partition is replicated in several brokers, if a broker fails the others have the necessary information for the system to continue working properly (Dobbelaere & Esmaili, 2017; *Kafka vs RabbitMQ - A Head-to-Head Comparison for 2023*, n.d.). Distributing the partitions among the brokers allows for increased performance. In the case of RabbitMQ a queue is used to replicate the messages. A Round-Robin algorithm is used to distribute messages between queues in order to increase their transfer without overloading the system. Several consumers can read messages from different queues at the same time (*Kafka vs RabbitMQ - A Head-to-Head Comparison for 2023*, n.d.). Kafka was designed to be horizontally scalable, that is, adding more devices while RabbitMQ was designed to be vertically scalable, that is, increasing its computational capacity (*Kafka vs RabbitMQ - A Head-to-Head Comparison for 2023*, n.d.).

Table 1 - Comparison between Apache Kafka and RabbitMQ

Feature	Apache Kafka	RabbitMQ
Architecture	Distributed, partitioned, replicated commit log service	Distributed message broker
Performance	High-throughput, low-latency, real-time data streaming	High-throughput, low-latency messaging
Source language	Written in Scala and Java	Written in Erlang
Push/Pull - Smart/Dumb	Pull-based, smart consumer model	Push-based, dumb consumer model
Scalability and Redundancy	Horizontally scalable, fault-tolerant, and highly available	Horizontally scalable, fault-tolerant, and highly available
Message Deletion	Based on time or size of the log	Based on consumption or time-to-live
Message Consumption	Consumer controls offset management and can replay messages	Consumer acknowledges messages after processing
Message Priority	No built-in support for message priority	Supports message priority

Sequential Ordering	Maintains order within a partition	Maintains order within a queue
Libraries and Language Support	Supports multiple languages through client libraries	Supports multiple languages through client libraries and plugins
Protocols Supported	Proprietary binary protocol over TCP/IP with support for SSL/TLS encryption and SASL authentication. Also supports HTTP/2 via Kafka REST Proxy.	Supports multiple protocols including AMQP, MQTT, STOMP, and HTTP/1.1 via plugins
Ease of use	East to use	Easy to use
Cost	Open source	Open source

Advantages of Apache Kafka:

- High throughput: Kafka can handle millions of messages per second.
- Horizontal scalability: Kafka can be scaled horizontally by adding more brokers.
- Durability: Messages in Kafka are persisted to disk, so they are not lost if a broker fails.
- Reliability: Kafka is a highly reliable system.
- Easy to use: Kafka is easy to use and has a simple API.

Advantages of RabbitMQ:

- Low latency: RabbitMQ has low latency, making it suitable for real-time messaging applications.
- Flexible routing: RabbitMQ supports a variety of message routing patterns.
- Easy to use: RabbitMQ is easy to use and has a simple API.

After analysing the main differences and advantages of each tool, Apache Kafka was chosen. The main reasons to choose Kafka are the following:

- Applications registered to receive messages from a topic are notified in real time as new messages arrive to Kafka broker. This feature allows applications that must quickly process available messages.
- The infrastructure communication can be easily scaled horizontally by adding more nodes to the cluster and partitions to individual topics.
- It can persist messages for a configurable period rather than deleting them as soon as they reach the destination application.
- It is a reliable platform with a huge user base.
- It allows the implementation of secure communications.

3.2 Kafka

Apache Kafka is an open-source platform for real-time data handling that enables the development of real-time, event-driven applications (*What Is Apache Kafka? | IBM*, n.d.). It is

a distributed streaming platform that can be used to collect, process, store, and integrate data at scale (*What Is Kafka, and How Does It Work? A Tutorial for Beginners*, n.d.). It has numerous use cases including distributed streaming, stream processing, data integration, and pub/sub messaging (*What Is Kafka, and How Does It Work? A Tutorial for Beginners*, n.d.).

Apache Kafka was created by LinkedIn to centralize the platform's real-time events (Goodhope et al., 2012). It has a large capacity and low latency in real-time data processing (Goodhope et al., 2012). Its focus is message streaming, and the result is a scalable system of transaction logs (Wang et al., 2015b).

Kafka has a mechanism where the producer is dumb and the consumers smart, that is, the producer puts the message in a certain topic and does not know if any consumer will receive the message. Here it is the consumer who manages the topics to which he subscribes, and messages received.

To promote scalability and redundancy Kafka relies on partitions. A partition is replicated in several brokers, if a broker fails the others have the necessary information for the system to continue working properly (Dobbelaere & Esmaili, 2017; *Kafka vs RabbitMQ - A Head-to-Head Comparison for 2023*, n.d.). Distributing the partitions among the brokers allows for increased performance.

Apache Kafka was chosen for this project because of is great capacity to deal with a lot of data at the same time in a fast and efficient way. Kafka also allows the creation of topics and if it is necessary to create more consumers for the topics it is relatively easy to do and for this project, this horizontal scalability is indicated.

3.2.1 Kafka Overall Architecture

The Figure 2 represents an example of Kafka's architecture. It is composed of zookeeper, broker (or server), client, producer, consumer, topic, partition, and message (*Apache Kafka*, n.d.)

- **Zookeeper**: According to the official documentation (*Apache ZooKeeper*, n.d.) it is "centralized service for maintaining configuration information, naming, providing distributed synchronization and group services.". A new version of Apache Kafka is being developed in which this component will disappear, and its functions will be carried out by the server.
- **Broker (server)**: A Kafka cluster can be made up of multiple brokers. These hold topics and the partitions within them, each of which can have partitions of multiple topics. One of the brokers in the cluster will be chosen as the leader and the others as followers. It is up to the leader to handle message transactions and the followers synchronize with the leader.
- **Client**: It allows the application to be distributed and use microservices to read, write and process data streams in parallel, at scale and with fault tolerance.
- **Producer**: Kafka client application that builds messages and publishes (writes) them. It is not dependent on any consumer.

- **Consumer**: Kafka client application that receives and reads messages published by producers. It is not dependent on the producers (it only needs them to receive messages) allowing the system greater scalability.
- **Topic**: One topic, and quoting the official Apache Kafka documentation, is “like a folder in a filesystem and the events are the files in that folder” (*Apache Kafka*, n.d.). A topic can be accessed (through its partitions) by several producers, or by none, in the same way that messages sent in that topic can be read by several, or no consumers. A topic and the messages in it can be read as many times as necessary because they are not deleted after each reading.
- **Partition**: Container that holds the data subsets of a topic. In the best case, each topic is divided into multiple partitions, and messages are sent cyclically between each partition.
- **Message**: It is the information that will circulate between producers and consumers, it consists of a key, a value and an offset. The messages must be linked to a topic so that the broker can choose the partition where they are placed.

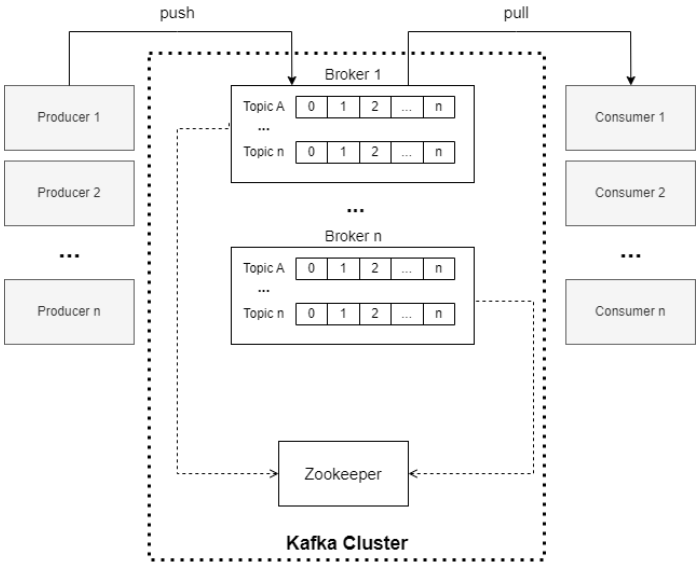

Figure 2 - Apache Kafka architecture example

The final architecture of the PLAYOFF project will be presented in Chapter 4.6.

3.2.2 Data protection in Apache Kafka

In addition to the project's ethical issues, the security of data circulating in the communication structure between MOG and ISEP is also of equal importance. Security is something that is extremely important nowadays, due to the range of cyber-attacks that occur or may occur on computer systems (Hu et al., 2021; Zambrano et al., 2023). To minimize the risks and effects of attacks, Kafka has 5 measures (*Apache Kafka - Security Overview*, n.d.):

1. Connections to brokers from clients (such as producers and consumers), other brokers, and tools can be authenticated using either SSL or SASL. Kafka supports several SASL mechanisms:
    a. SASL/GSSAPI (Kerberos)
    b. SASL/PLAIN
    c. SASL/SCRAM-SHA-256 and SASL+/SCRAM-SHA-512
    d. SASL/OAUTHBEARER

2. Connections from brokers to Zookeeper can be authenticated.
3. Data transferred between brokers and clients, between brokers, or between brokers and tools can be encrypted using SSL.
4. Clients can have authorizations for read/write operations.
5. Authorization is customizable and can be integrated with external authorization services.

The security layer in Kafka is optional and these 5 measures can be used individually or together (*Apache Kafka - Security Overview*, n.d.).

Apache Kafka uses PLAINTEXT by default and for encrypt the communication is necessary configure TLS/SSL. TLS is the newer version of SSL, it is more secure, has more features and stronger protection than SSL (*TLS vs. SSL: What's the Difference? - Rublon*, 2023).

Due to the nature of the project, only SASL/PLAIN is being used, which is the lowest level of security. Other measures involving authentication and authorization were also tested, but due to the complexity they would add to a project of an academic nature and low risk in terms of possible attacks, it was set aside for a later phase and is referenced for future work. and improvement to what has been developed so far.

3.3 Ethical Issues in AI

With the growth in the use and capabilities of AI (Lu, 2019), it is necessary to take into account the dangers that it can bring, such as the disclosure of personal information, increased inequality between social classes and even environmental pollution, for example. At the same time, it raises several ethical issues in human rights, information, and responsibility. In order to mitigate these negative aspects, or even make them non-existent, a strong international collaboration of all governments is required, with measures on the use of AI (*Strengthening International Cooperation on AI | Brookings*, n.d.; Zuiderwijk et al., 2021). In the GDPR and despite the impact that AI has on privacy, it allows some exceptions to other rules described in it to be ignored for research purposes. These exceptions should not happen except in cases where there is public interest in the projects, and they would have to be validated by independent ethics committees (Lorè et al., 2023; Meszaros & Ho, 2021; van Bekkum & Zuiderveen Borgesius, 2023). This is the only way we can ensure that the rules are complied

with and that the secrecy and low ethical standards of the companies do not compromise the data used.

There are five pillars extremely important to be able to apply ethical governance in a company or organization. These pillars can be summarized as follows (Benjamins, 2020; Cihon et al., 2021; Ibáñez & Olmeda, 2022; Mäntymäki et al., 2022):

1. Publish a code of ethical conduct.
2. Provide training on ethics and responsible innovation for everyone.
3. Practice responsible innovation by involving stakeholders. This way, all new products will be analysed if they correspond to the intended ethical criteria.
4. Be transparent regarding ethical governance.
5. Value ethical governance and respect all the criteria defined in it.

Several AI fields have used information from social networks to perform processing and analysis on a wide range of topic. When using AI techniques on data retrieved from social networks, it is necessary to respect ethical guidelines (Taylor & Pagliari, 2018). These guidelines are:

- **Respect the privacy and consent of social media users**: Avoid analysing sensitive information and always respect the terms and conditions of the platform in question. Use the data anonymously, that is, that the information collected is not associated with a particular user but aggregated in a dataset of several users.
- **Minimize harm or potential risks to social media users**: Investigators must identify and mitigate the risk of emotional distress, reputational damage, discrimination, or manipulation. They must also monitor and report any adverse events that occur during and/or after the research.
- **Ensure the quality and validity of data and analysis**: Researchers must use rigorous methods and the right tools to obtain social media data, as well as to analyse that data. They also must accept that the data may have limitations and, above all, may be biased and therefore the results must be well analysed, otherwise incorrect and misleading conclusions may occur.
- **Increase the transparency and accountability of investigations**: Researchers must disclose the purpose, methods, sources, and results of their research to all interested parties, such as sponsors, social media platforms, users and ethics committees. Data and code should be released for verification and reuse whenever possible.

To these guidelines can also be added, for analysis of sentiment and topics, the following:

- **Respect the emoticons and opinions of users**: The vision and interpretation of the researchers cannot be put above the feelings and opinions of the users. Feelings and affairs cannot be manipulated either (Drus & Khalid, 2019; Wankhade et al., 2022).

4 Communication Infrastructure

One of the requirements of this project was to develop a communication structure between the servers of ISEP and the servers of MOG. But before starting the study of the best possible architecture for this, it was necessary to choose the communication framework used. In the chapter 3 this choice is described and the reasons why Apache Kafka was chosen.

Something that needed to be taken into consideration when building this tool was the connection to the APIs of social networks, Twitter, and Reddit, in the original version and YouTube in the most recent version.

In this project Apache Kafka was used and hosted directly on the server (Windows Server 2012) instead of using a Docker.

4.1 Kafka Configuration and Commands

To use Apache Kafka in the project was necessary to configure it according to the specifications of the ISEP server.

Despite several versions of Kafka being used throughout the development of the project, the configuration remained the same and it was not necessary to change any parameters and the creation of topics also always followed the same command line commands. The following subchapters refer to the files that need to be changed in Apache Kafka and the configurations used, as well as the commands used to run the zookeeper and broker and in the creation of Kafka topics.

4.1.1 Configurations

Apache Kafka is composed of several folders and files. The configuration files are inside a folder called “configs”. This is where the documents necessary to configure our project are located, they are “zookeeper.properties”, “server.properties”.

4.1.1.1 Properties for Zookeeper

- dataDir=./data/zookeeper
- clientPort=2181
- maxClientCnxns=0
- admin.enableServer=false

4.1.1.2 Properties for Server

- broker.id=0
- listeners= PLAINTEXT://192.168.2.137:9092
- advertised.listeners= PLAINTEXT://playoff.gecad.isep.ipp.pt:9092
- listener.security.protocol.map=PLAINTEXT:PLAINTEXT
- num.network.threads=3
- num.io.threads=8
- socket.send.buffer.bytes=102400
- socket.receive.buffer.bytes=102400
- socket.request.max.bytes=104857600
- log.dirs=./data/server
- num.partitions=3
- num.recovery.threads.per.data.dir=1
- offsets.topic.replication.factor=1
- transaction.state.log.replication.factor=1
- transaction.state.log.min.isr=1
- log.retention.hours=168
- log.retention.check.interval.ms=300000
- zookeeper.connect=192.168.2.137:2181
- zookeeper.connection.timeout.ms=18000
- group.initial.rebalance.delay.ms=0

4.1.2 Commands

All these commands are to run on Command Prompt

- Start the Zookeeper:

```
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

- Start the Server(broker):

```
.\bin\windows\kafka-server-start.bat .\config\server.properties
```

- Create a Kafka topic:

```
.\bin\windows\kafka-topics.bat --create --topic {topic_name} --
bootstrap-server localhost:9092
```

- List topics:

```
.\bin\windows\kafka-topics.bat --list --bootstrap-server
localhost:9092Error! Reference source not found.
```

4.2 Kafka Topics

As indicated in the chapter 3.2, the communication system using Kafka requires topics for the producer to publish messages on a topic and the consumer to read messages from a topic.

4.2.1 Topics for messages from MOG to ISEP

- "master_topic"

The "master_topic", is the topic to which the MOG producer makes all analysis requests, and which will be received by the ISEP 'master' consumer. The format of the messages in this topic is described in the chapter 4**Error! Reference source not found.**.

4.2.2 Topics for messages from ISEP to MOG

- "twitter_SA"
- "twitter_TA"
- "reddit_SA"
- "reddit_TA"
- "chat_SA"
- "chat_TA"
- "feedback"

All the topics that are used by the ISEP producer to send messages to MOG indicate the social network from which the data was taken and what type of analysis. All topics with "SA" in their name are topics that indicate Sentiment Analysis, all those with "TA" are Topic Analysis topics.

These topics were chosen to make it possible to identify the social network and the type of analysis performed. Using this approach also allows, if desired, to perform only one of the analyses on a particular social network, without interference with the others. These topics also serve for the ISEP "master" consumer to identify the analysis and social network to be executed, so it is described in one of the fields of the MOG to ISEP messages.

There is also another topic, called “feedback”, this one was only introduced in the last approach to confirm that ISEP received the message from MOG, as it will be described in 4.6.2.2.

Later in the project the topics “twitter_SA” and “twitter_TA”, were replaced by “youtube_SA” and “youtube_TA”, when the change to the social media chosen was made.

4.3 First Approach

Here, the architecture and the multiple messages formats option to be considered of the communication structure are displayed.

4.3.1 Architecture

This approach was not implemented, but it was the starting point for the remaining approaches and implementations. In it, the main points of the project were already defined: Two servers, one from ISEP, another from MOG. On the MOG server there would be one or more consumers, who would receive the response messages coming from ISEP, after the NLP analyses were performed. The core components of Kafka, that is, the broker and zookeeper are on the ISEP server and managed from there. In Figure 3 is the draft of this approach.

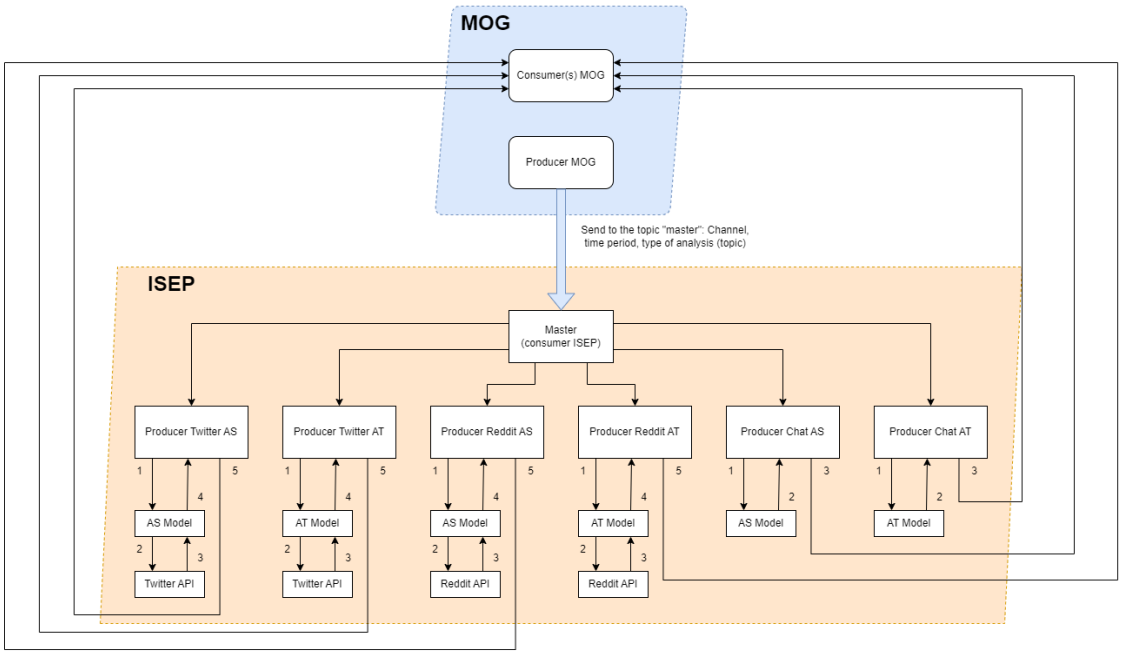

Figure 3 - First architecture draft of the Kafka project

It was also defined that the Kafka producer that originates the entire execution cycle would be on the MOG side, that is, when a user of the platform developed by MOG requests an analysis of an event, the start message of analysis will be sent by this producer to the ISEP consumer.

At this stage, it was also necessary to choose the format of the messages that would be exchanged by Kafka.

On the ISEP server, there is a consumer, designated as the "master_consumer" for being the brain of the entire operation. This is responsible for receiving messages from the MOG producer and only checking which social network to analyse and what type of analysis, then it would call the code from the producer's file, and this would do all the corresponding processing.

Due to the different format of the information processed and returned by the sentiment analysis and topic analysis models (Ferraz Barbosa Soares de Albergaria, 2022), it was necessary for the format of the messages to be different from each other. Thus, in this first draft, each social network and each type of analysis would have a different producer, and this would be responsible for processing the data from the request message coming from MOG. In this, the API of the corresponding social network would be called in order to obtain the data (comments and posts) of the desired event on that social network, then this data would be placed in the correct format for the ML model and sent to that model, that is, an instance of that model would be called with the data obtained. All text pre-processing is done in the models, so it is not necessary to take this into account in this system. After processing by the model, the result of the analysis would be returned to the producer, and it would create a Kafka message in json format where it would have information on the result of the analysis and send it to MOG.

It should be noted that in this approach the system was designed so that each time an analysis was needed, even if it was the same event, the MOG producer would send a message to the "master_consumer" indicating that it wanted a new analysis. For example, if there were four events at the same time and analyses of those events from both social networks and the Chat Platform comments were desired, six messages per event would be required for just one analysis. If analyses were desired every five minutes for two hours, there would be twenty-five messages per event, which corresponds to one hundred request messages in the MOG to ISEP direction and another one hundred response messages in the ISEP to MOG direction.

4.3.2 Format of the messages

In the beginning, before the format of the messages was decided, three hypotheses of request message format were studied, that is, the messages that MOG sends to ISEP to initiate the analyses. Depending on what was decided between these three options, the format of the response message from ISEP to MOG would be elaborated. In the project there are two types of messages. The ones that MOG send to ISEP with the request and the ones that ISEP send to MOG with the responses.

The file format of the messages exchanged by Kafka is JSON[18].

4.3.2.1 Option 1

```
{
    "timestamp": "2012-04-23T18:25:43.511Z",
    "channel": "",
    "time_period": 90,
    "topic": [
        "chat_TA"
    ],
    "chat_messages": [
        "msg1",
        "msg2"
    ]
}
```

Code 1 - Example of a Kafka message to analyse Chat Platform comments in the first option.

This type of messages is always sent to the Kafka topic named “master_topic” and has the following parameters:

- “timestamp”: timestamp of the request. Format of the timestamp in this option, “2012-04-23T18:25:43.511Z”.
- “channel”: channel to be analysed. Example of a twitter channel, “#portosporting”. Example of a reddit channel, “https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp_portuguese/”.
- “time_period”: The number of minutes before the timestamp of the message to fetch comments.
- “kafka_topics”: Kafka topic corresponding to the type of analysis and the intended social media. The topic can be:
    - “reddit_TA”: indicate that is to do a topic analysis of reddit information.
    - “reddit_SA”: indicate that is to do a sentiment analysis of reddit information.
    - “twitter_TA”: indicate that is to do a topic analysis of twitter information.
    - “twitter_SA”: indicate that is to do a sentiment analysis of twitter information.
    - “chat_TA”: indicate that is to do a topic analysis of Chat Platform comments.
    - “chat_SA” indicate that is to do a sentiment analysis of Chat Platform comments.
- “chat_messages”: Chat Platform comments to be analysed. They are inside an array.

In this type of message, for each social network and desired NLP analysis, it would be necessary to send a request message, that is, to analyse the sentiments and topics of an event on all platforms, it is necessary to send six messages per event. This format is the simplest and allows for a lower degree of difficulty in programming, but the number of messages is large.

[18] https://www.json.org/json-en.html

It is composed of the sending timestamp, the channel that corresponds to the reddit URL or Twitter hashtag, the topic of the message, which indicates the analysis and platform to be analysed. In the case of chat, there is a field where the Chat Platform comments to be analysed will be sent.

4.3.2.2 Option 2

```
{
    "timestamp": "2012-04-23T18:25:43.511Z",
    "channel": "",
    "time_period": 90,
    "topics": [
        "chat_TA",
        "chat_SA"
    ],
    "chat_messages": [
        "msg1",
        "msg2"
    ]
}
```

Code 2 - Example of a Kafka message to analyse Chat Platform comments in the second option.

This type of messages has the following parameters:

- “timestamp”: timestamp of the request. Format of the timestamp in this option, “2012-04-23T18:25:43.511Z”.
- “channel”: channel to be analysed. Example of a twitter channel, “#portosporting”. Example of a reddit channel, “https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp_portuguese/”.
- “time_period”: The number of minutes before the timestamp of the message to fetch comments.
- “kafka_topics”: Kafka topic corresponding to the type of analysis and the intended social media. The topic can be:
    - “reddit_TA”: indicate that is to do a topic analysis of reddit information.
    - “reddit_SA”: indicate that is to do a sentiment analysis of reddit information.
    - “twitter_TA”: indicate that is to do a topic analysis of twitter information.
    - “twitter_SA”: indicate that is to do a sentiment analysis of twitter information.
    - “chat_TA”: indicate that is to do a topic analysis of Chat Platform comments.
    - “chat_SA” indicate that is to do a sentiment analysis of Chat Platform comments.
- “chat_messages”: Chat Platform comments to be analysed. They are inside an array.

In option 2, each message would correspond to a social network, so if the intention is to perform the analyses on the three platforms (reddit, twitter and chat), three messages would be necessary, but in the case of being to perform sentiment analysis and topic analysis, this indication would come in the message and an extra message would not be necessary. In this way it is also possible to indicate only one of the analyses. For example, it is possible to request both analyses for the Twitter social network, but only topic analysis for the reddit social network and sentiment analysis for Chat Platform. This format allows for greater versatility without requiring an excessive number of messages, at the programming level it requires greater complexity compared to **Error! Reference source not found.**

Each message is composed of the timestamp of its sending, the channel that corresponds to the URL of the event in question on the reddit social network, or to the hashtag on the Twitter social network. If the option was to analyse the comments from the internal platform chat, then this field would come empty.

4.3.2.3 Option 3

```
{
    "timestamp": "2012-04-23T18:25:43.511Z",
    "time_period": 90,
    "channels": [
        {
            "channel":
"https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp
_portuguese/",
            "topics": [
                "reddit_TA",
                "reddit_SA"
            ]
        },
        {
            "channel": "#portosporting",
            "topics": [
                "twitter_TA",
                "twitter_SA"
            ]
        },
        {
            "channel": "chat",
            "topics": [
                "chat_TA",
                "chat_SA"
            ],
            "chat_messages": [
                "msg1",
                "msg2",
```

```
                "msg3"
            ]
        }
    ]
}
```

Code 3 - Example of a Kafka message in the third option.

This type of messages is always sent to the Kafka topic named “master_topic” and has the following parameters:

- “timestamp”: timestamp of the request. Format of the timestamp in this option, “2012-04-23T18:25:43.511Z”.
- “time_period”: The number of minutes before the timestamp of the message to fetch comments.
- “channels”: channels to be analysed. Also correspond to the social networks.
    - “channel”: channel to be analysed. Example of a twitter channel, “#portosporting”. Example of a reddit channel, “https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp_portuguese/”.
    - “kafka_topics”: Kafka topic corresponding to the type of analysis and the intended social media. The topic can be:
        - “reddit_TA”: indicate that is to do a topic analysis of reddit information.
        - “reddit_SA”: indicate that is to do a sentiment analysis of reddit information.
        - “twitter_TA”: indicate that is to do a topic analysis of twitter information.
        - “twitter_SA”: indicate that is to do a sentiment analysis of twitter information.
        - “chat_TA”: indicate that is to do a topic analysis of Chat Platform comments.
        - “chat_SA” indicate that is to do a sentiment analysis of Chat Platform comments.
    - “chat_messages”: Chat Platform comments to be analysed. They are inside an array.

Option 3 is the most complex. In this case, each event corresponds to a message. In this message, in addition to the timestamp, several channels are indicated (three at most), one for each platform to be analysed. Each social network corresponds to an element within the “channels” array, where the channel of that social network and the analyses to be carried out on that channel are indicated. It is the best version for the number of messages exchanged, but the most complex to implement.

4.4 Second Approach

In this chapter, the architecture and the messages formats of the second approach of the communication structure are displayed.

In this approach the Kafka version used was 3.3.1 [19]

4.4.1 Architecture

This second approach is based on the first architecture described in 4.3.1, but it has already been implemented and already had a defined message structure. The Figure 4 represents the implementation.

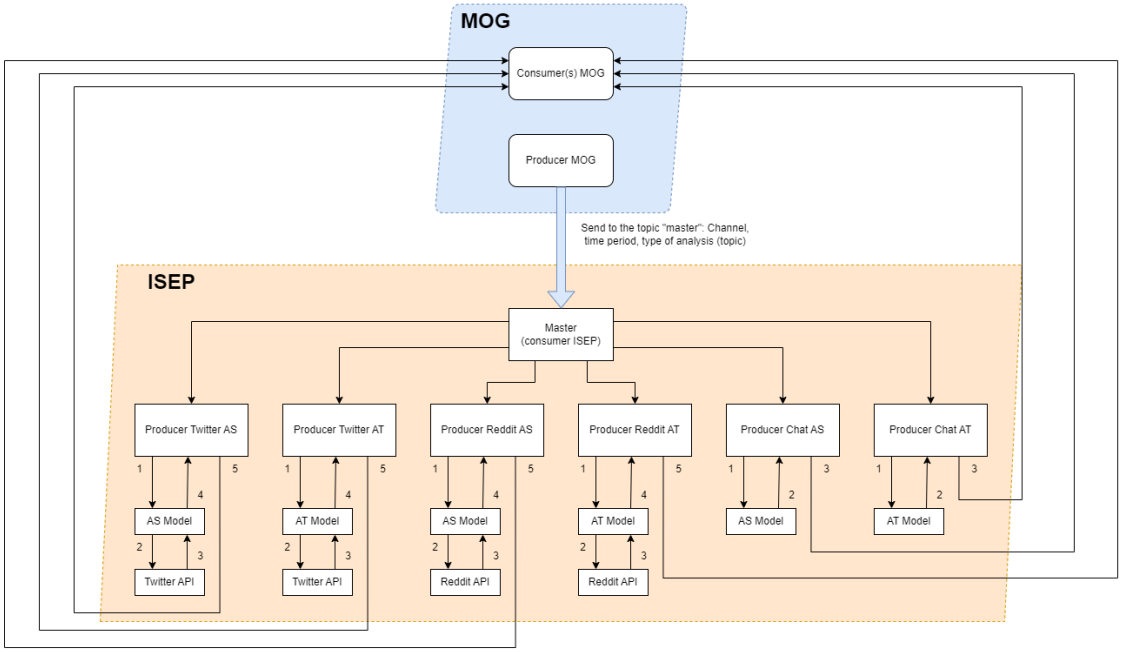

Figure 4 - Second architecture of the Kafka project

4.4.2 Format of the messages

The type of messages chosen and defined for the rest of the project was Option 2 described in 4.3.2.2. But this underwent a small change, as can be seen below.

4.4.2.1 Messages from MOG to ISEP

This type of messages has the following parameters:

- "timestamp": timestamp of the request. Format of the timestamp in this option, "2012-04-23T18:25:43.511Z".

[19] https://archive.apache.org/dist/kafka/3.3.1/kafka-3.3.1-src.tgz

- “channel”: channel to be analysed. Example of a twitter channel, “#portosporting”. Example of a reddit channel, “https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp_portuguese/”.
- “init_time”: The time instant of the first comment to be analysed. Format of the timestamp in this option, “2012-04-23T18:25:43.511Z”.
- “end_time”: The time instant of the last comment to be analysed. Format of the timestamp in this option, “2012-04-23T18:25:43.511Z”.
- “kafka_topics”: Kafka topic corresponding to the type of analysis and the intended social media. The topic can be:
    - “reddit_TA”: indicate that is to do a topic analysis of reddit information.
    - “reddit_SA”: indicate that is to do a sentiment analysis of reddit information.
    - “twitter_TA”: indicate that is to do a topic analysis of twitter information.
    - “twitter_SA”: indicate that is to do a sentiment analysis of twitter information.
    - “chat_TA”: indicate that is to do a topic analysis of Chat Platform comments.
    - “chat_SA” indicate that is to do a sentiment analysis of Chat Platform comments.
- “chat_messages”: Chat Platform comments to be analysed. They are inside an array.

The visible change in this approach is the elimination of the ‘time_period’ field and the inclusion of two fields, designated as ‘init_time’ and ‘end_time’. The first serves to indicate the time of the first comment to be analysed and the second indicates the time of the last comment. In this way, it is ensured that all comments between ‘init_time’ and ‘end_time’ are analysed.

4.4.2.2 Messages from ISEP to MOG

Sentiment Analysis Messages

```
{
    "timestamp": "2012-04-23T18:30:43.511Z",
    "request_timestamp": "2012-04-23T18:25:43.511Z",
    "channel":
"https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp
_portuguese/",
    "emotions": {
        "NEGATIVE": 0.3,
        "NEUTRAL": 0.6,
        "POSITIVE": 0.1
    },
    "n": 934
}
```

Code 4 - Example of a Kafka message with the results of the sentiment analysis in the second approach.

This type of messages has the following parameters:

- “timestamp”: timestamp of the message.
- “request_timestamp”: timestamp of the request.
- “channel”: channel to be analysed.
- “emotions”: object that have all the sentiments possible in the analysis and their respective value. The sentiments can be:
    - “NEGATIVE”: percentage of negative comments
    - “NEUTRAL”: percentage of neutral comments
    - “POSITIVE”: percentage of positive comments
- “n”: number of comments that where analysed.

Topic Analysis Messages

```
{
    "timestamp": "2012-04-23T18:30:44.511Z",
    "request_timestamp": "2012-04-23T18:25:43.511Z",
    "channel":
"https://www.reddit.com/r/soccer/comments/sq8rai/match_thread_fc_porto_vs_sporting_cp
_portuguese/",
    "topics": {
        "save": 0.05,
        "foul": 0.05,
        "free_kick": 0.1,
        "goal": 0.1,
        "kickoff": 0.1,
        "offside": 0.1,
        "penalty": 0.1,
        "red_card": 0.1,
        "yellow_card": 0.1,
        "back_pass": 0.1,
        "substitution": 0.1
    },
    "n": 934
}
```

Code 5 - Example of a Kafka message with the results of the topic analysis in the second approach.

This type of messages has the following parameters:

- “timestamp”: timestamp of the message.
- “request_timestamp”: timestamp of the request.
- “channel”: channel to be analysed.
- “topics”: object that have all the topics possible in the analysis and their respective value. The topics can be:
    - “save”: percentage of comments that have ‘save’ has the main topic.

    - “foul”: percentage of comments that have ‘foul’ has the main topic.
    - “free_kick”: percentage of comments that have ‘free_kick’ has the main topic.
    - “goal”: percentage of comments that have ‘goal’ has the main topic.
    - “kickoff”: percentage of comments that have ‘kickoff’ has the main topic.
    - “offside”: percentage of comments that have ‘offside’ has the main topic.
    - “penalty”: percentage of comments that have ‘penalty’ has the main topic.
    - “red_card”: percentage of comments that have ‘red_card’ has the main topic.
    - “yellow_card”: percentage of comments that have ‘yellow_card’ has the main topic.
    - “back_pass”: percentage of comments that have ‘back_pass’ has the main topic.
    - “substitution”: percentage of comments that have ‘substitution’ has the main topic.
- “n”: number of comments that where analysed.

This type of message has the result of sentiment and topic analyses in the ‘emotions’ and ‘topics’ fields, respectively. It also has a field designated as ‘n’, which corresponds to the number of comments analysed to obtain the values referred to in ‘emotions’ or ‘topics’. The ‘request_timestamp’ field indicates the moment in time when the request was made by MOG and the ‘timestamp’ is the moment when this message was sent. The values assigned to each sentiment or topic correspond to the percentage of comments that have that sentiment or topic as predominant.

4.5 Third Approach

In this chapter, the architecture and the messages formats of the third approach of the communication structure are displayed.

In this approach the Kafka version used was 3.3.2[20]

4.5.1 Architecture

Based on the architecture represented in Figure 3 and Figure 4, the Figure 5 shows the second system architecture. The system is composed of two distinct parts. One corresponding to MOG servers and the other to ISEP servers. On MOG’s side, there is a Kafka producer and a Kafka consumer. On ISEP’s side, there is a consumer, designated as master, and a producer that will send the analysis results to MOG. Within the ISEP side are the sentiment analysis and topic analysis models. On the MOG side, there is the front-end platform, where the user will have to interact. The author is only responsible for the ISEP part.

[20] https://archive.apache.org/dist/kafka/3.3.2/kafka_2.13-3.3.2.tgz

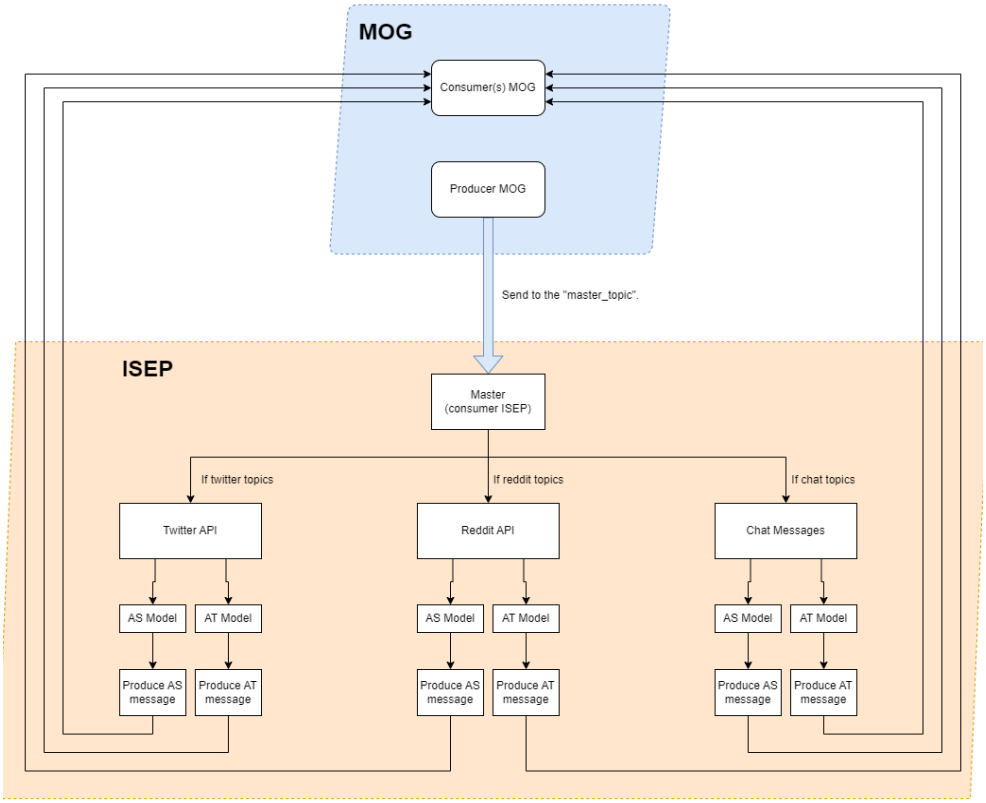

Figure 5 - Third architecture of the Kafka project

4.5.2 Format of the messages

4.5.2.1 Messages from MOG to ISEP

This type of messages is always sent to the Kafka topic named “master_topic” and has the following parameters:

- “event_id”: id of the event.
- “timestamp”: timestamp of the request in Unix. An example of the timestamp’s format in this approach is “1680570542”.
- “channel”: channel to be analysed.
- “analysis_status”: status of the analysis, in other words, if is to start or end the analysis of that event. This can be:
    - “start”: indicate that is to start an analysis of the event.
    - “end”: indicate that is to terminate an analysis of the event.
- “kafka_topics”: Kafka topic corresponding to the type of analysis and the intended social media. The topic can be:
    - “reddit_TA”: indicate that is to do a topic analysis of reddit information.
    - “reddit_SA”: indicate that is to do a sentiment analysis of reddit information.
    - “twitter_TA”: indicate that is to do a topic analysis of twitter information.

    - "twitter_SA": indicate that is to do a sentiment analysis of twitter information.
    - "chat_TA": indicate that is to do a topic analysis of Chat Platform comments.
    - "chat_SA" indicate that is to do a sentiment analysis of Chat Platform comments.
- "chat_messages": Chat Platform comments to be analysed.

The main change in this approach was the introduction of the "analysis_status" field and removal of the "init_time" and "end_time". If was to start an analysis this would be "start" and if it was to terminate that analysis it would be "end". This change originated that for every event there will be two messages from MOG to ISEP and between them all the messages from ISEP to MOG with the results of the analysis. This also means that the comments to analysed will always be inside the time frame of the two messages, so old messages, older than the message of the "start" messages don't matter.

It is also possible to verify a change in the format of the 'timestamp' field. This has changed from ISO 8601 to Unix timestamp. This exchange occurred to facilitate the obtaining of data in the reddit API, as described in the chapter 4.7.

The topic indicates in the "kafka_topics" parameter, can only be for one platform, which means that if it is necessary to analyse an event on reddit, twitter and Chat Platform, three messages are required to initiate these analyses, one for each analysis.

In case of the information to be analysed be messages from the Chat Platform, the parameter "channel" should be empty because there is no data to be obtained by social media APIs, or vice versa.

4.5.2.2 Messages from ISEP to MOG

<u>Sentiment Analysis parameters:</u>

This type of messages has the following parameters:

- "event_id": the id of the event.
- "timestamp": timestamp of the message in Unix.
- "request_timestamp": timestamp of the request.
- "channel": channel to be analysed.
- "emotions": object that have all the sentiments possible in the analysis and their respective value. The sentiments can be:
    - "NEGATIVE": percentage of negative comments
    - "NEUTRAL": percentage of neutral comments
    - "POSITIVE": percentage of positive comments
- "n": number of comments that where analysed.

Topic Analysis parameters:

This type of messages has the following parameters:

- "event_id": the id of the event.
- "timestamp": timestamp of the message in Unix.
- "channel": channel to be analysed.
- "topics": object that have all the topics possible in the analysis and their respective value. The topics can be:
    - "save": percentage of comments that have 'save' has the main topic.
    - "foul": percentage of comments that have 'foul' has the main topic.
    - "free_kick": percentage of comments that have 'free_kick' has the main topic.
    - "goal": percentage of comments that have 'goal' has the main topic.
    - "kickoff": percentage of comments that have 'kickoff' has the main topic.
    - "offside": percentage of comments that have 'offside' has the main topic.
    - "penalty": percentage of comments that have 'penalty' has the main topic.
    - "red_card": percentage of comments that have 'red_card' has the main topic.
    - "yellow_card": percentage of comments that have 'yellow_card' has the main topic.
    - "back_pass": percentage of comments that have 'back_pass' has the main topic.
    - "substitution": percentage of comments that have 'substitution' has the main topic.
- "n": number of comments that where analysed.

As in the second approach, the 'timestamp' field is in Unix format and the values presented for each sentiment or topic correspond to the percentage of comments that have that sentiment or topic as predominant.

4.6 Fourth Approach (final)

In this chapter, the architecture and the messages formats of the final approach of the communication structure are displayed.

In this approach the Kafka version used was 3.4.0[21]

4.6.1 Architecture

The overall architecture of the final version is the same as the previous approach, only changing the social networks used, as represented in the Figure 6 and also changes and improvements were made in the implementation of the code. These changes are described in sub-chapter 4.8.

[21] https://archive.apache.org/dist/kafka/3.4.0/kafka_2.13-3.4.0.tgz

The system consists of two parts, the MOG server and the ISEP server. They communicate with each other through Kafka messages, using pre-defined topics (see chapter 4.2).

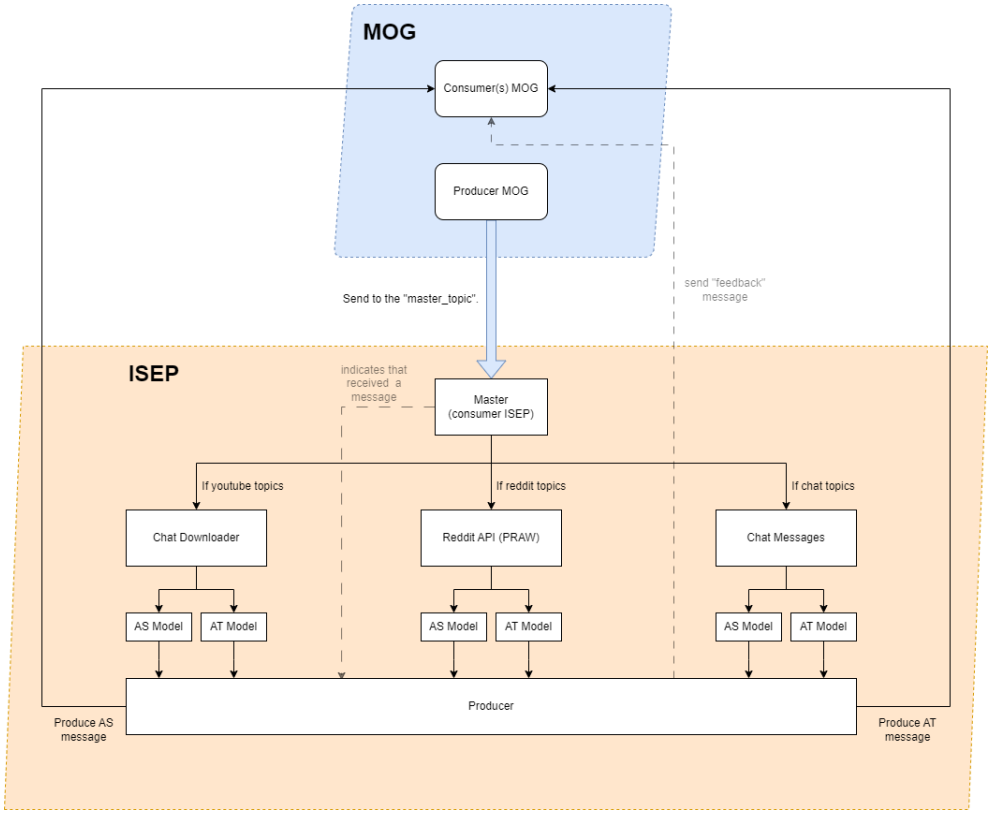

Figure 6 - Final Kafka Architecture

4.6.2 Format of the messages

4.6.2.1 Messages from MOG to ISEP

This type of message is the same as the Third Approach, only changing the topics used because the social networks have changed, so it has the following fields:

- "event_id": id of the event.
- "timestamp": timestamp in Unix of the request.
- "channel": channel to be analysed.
- "limit": the maximum amount of comments that will be analyse in an iteration.
- "analysis_status": status of the analysis, in other words, if is to start or end the analysis of that event. It can be:
    - "start".
    - "end".
- "kafka_topics": Kafka topic corresponding to the type of analysis and the intended social media. The topic can be:

    - "reddit_TA": indicate that is to do a topic analysis of reddit information.
    - "reddit_SA": indicate that is to do a sentiment analysis of reddit information.
    - "youtube_TA": indicate that is to do a topic analysis of YouTube information.
    - "youtube_SA": indicate that is to do a sentiment analysis of YouTube information.
    - "chat_TA": indicate that is to do a topic analysis of Chat Platform comments.
    - "chat_SA" indicate that is to do a sentiment analysis of Chat Platform comments.
- "chat_messages": Chat Platform comments to be analysed.

Almost the entire body of the messages is the same as the messages from the previous approach, so the operation and meanings of the fields are the same. The only difference is the introduction of the "limit" parameter. This indicate the max amount of comments to be analysed in each iteration. If not specified, the default amount is 200. For example, if the data array to be analyse has more than this amount, there will a loop until all the comments are analysed. This was designed this way, to help performance in NLP models.

4.6.2.2 Messages from ISEP to MOG

This type of messages has two different forms, one for the sentiment analysis and other for the topic analysis.

<u>Sentiment Analysis parameters:</u>

This type of messages has the following parameters:

- "event_id": the identifier of the event.
- "timestamp": timestamp of the message in Unix.
- "channel": channel to be analysed.
- "emotions": object that have all the sentiments possible in the analysis and their respective value. The sentiments can be:
    - "NEGATIVE": value of negative comments
    - "NEUTRAL": value of neutral comments
    - "POSITIVE": value of positive comments
- "n": number of comments that where analysed.

In the format message there were no changes, but in the meaning of the values inside the "emotions" there were. This values instead of the percentage; they now represent the average of each sentiment. That is, every comment has values for the three sentiments, but in the previous approaches only the predominant value was considered, even if the difference between it be negative or neutral was minimal. This way every value of every comment has been considered and the result are way better. This was one of the changes required to the models.

Topic Analysis parameters:

This type of messages has the following parameters:

- “event_id”: the id of the event.
- “timestamp”: timestamp of the message in Unix.
- “channel”: channel to be analysed.
- “topics”: object that have all the topics possible in the analysis and their respective value. The topics can be:
    - “save”: value of comments that have ‘save’ has the main topic.
    - “foul”: value of comments that have ‘foul’ has the main topic.
    - “free_kick”: value of comments that have ‘free_kick’ has the main topic.
    - “goal”: value of comments that have ‘goal’ has the main topic.
    - “kickoff: value of comments that have ‘kickoff’ has the main topic.
    - “offside”: value of comments that have ‘offside’ has the main topic.
    - “penalty”: value of comments that have ‘penalty’ has the main topic.
    - “red_card”: value of comments that have ‘red_card’ has the main topic.
    - “yellow_card”: value of comments that have ‘yellow_card’ has the main topic.
    - “back_pass”: value of comments that have ‘back_pass’ has the main topic.
    - “substitution”: value of comments that have ‘substitution’ has the main topic.
- “n”: number of comments that where analysed.

Like in the sentiment analysis, this messages now represent the real value of each topic.

In this case, the messages are similar to those of Third Approach, with the addition of the “request_timestamp” field, which indicates the moment when the request, that is, the message from MOG was sent indicating that they wanted an analysis. In this version, the values also stopped corresponding to the percentages of comments with a certain sentiment or topic, but rather to certainty. That is, how certain is the model that a batch of comments had those sentiments and those topics. These new workings of the model’s results are explained in detail in chapter 4.8.

Feedback message:

There is a third type of messages in this approach, the feedback message. This one serve to ISEP confirm MOG that has received the message to start or finish an analysis request. In the previous approaches this didn’t happen, and in the case of some lost messages there was no way to know.

This type of messages has the following parameters:

- “event_id”: the identifier of the event.
- “channel”: channel to be analysed.
- “request_timestamp”: timestamp of the request.
- “result”: this is always “received”, to confirm the arrival of the message.
- “Type”:
    - “start”: to indicate that the received message is to start the analysis.

- "end": to indicate that the received message is to end the analysis.

4.7 Retrieve APIs data

4.7.1 Reddit

In order to obtain the comments of the intended events, it was necessary to connect to the Reddit API. The tool PRAW was used to make this connection. This tool is a python library that provides simple access to Reddit's API. It is designed to be user-friendly and interact with Reddit allowing to make post, commenting on an existing post, responding to a comment, fetching the list of moderators, get subreddits inside a reddit topic, etc.

The process of using PRAW and get the comments of a specific URL involved several steps, which I will outline below.

Firstly, I installed PRAW using pip, python's package installer. This was done by running the command `pip install praw` in the terminal.

Once PRAW was installed, I needed to authenticate my application with Reddit. This involved creating a script application on Reddit's website to obtain a client_id, client_secret, and user_agent. These credentials were then used to instantiate my Reddit instance in my python script. In addition to these credentials, it was also necessary to use the username and password of the Reddit account used to create the script application on the site.

With the Reddit instance ready, I could access a variety of data from Reddit. For this project, my goal was to fetch comments from a specific event via URL. To do this, I created a submission instance using the URL of the event, this allowed me to access the comments tree, which included all top-level comments and their respective replies. For this project, all of those comments were treated the same way as they were all top-level comments.

To fetch all comments from the event, I used the `replace_more` method on the submission's comments to get all the comments. Then, I iterated over the comment tree using a for loop and added each comment's body to an array called `data`. This array was the one to send to the inference of the sentiment analysis and topic analysis models. Before adding a comment to the array, a comparison was made between the creation date of each comment and the date of the last comment added in the last request to Reddit. This way, there were no duplicate comments. This comparison was necessary because the Reddit API has a limitation in which it does not allow indicating the time interval in which the comments were created, nor does it allow be always connected and automatically get the comments every time on was send to the post. There was a little difference between the thread and the multiprocessing approach of the way data from Reddit was obtained and added to the array as described in 4.8.

During this procedure, it was crucial to respect Reddit's terms of service and utilize their API in a responsible manner. This involved avoiding inundating their servers with excessive simultaneous requests.

4.7.2 YouTube

In order to obtain the comments/messages from YouTube I used Chat Downloader[22]. This is a simple tool that obtains chat messages from live streams in YouTube (also works with Twitch and Zoom), so is perfect for this project, and don't require to use the official YouTube API[23]. It parses chat items/messages into JSON objects, which include fields like message_id, message, message_type, timestamp, time_in_seconds, time_text, and author. For this project only the message text was considered and used.

The first step was to install the tool, to do this was used the command `pip install chat-downloader`. After installing, it was not necessary to configure anything specific, nor use any type of authentication unlike what was done with PRAW for Reddit.

Unlike PRAW, Chat Downloader allows for a continuous connection and whenever a comment is added to the live stream, it is automatically obtained. To make the connection, it is necessary to use the `get_chat` method of the tool, where the URL is indicated and subsequently a `for` loop to obtain the comments from it.

As the comments were being obtain in real time, it was not necessary to perform any type of date comparison to add each one of them to the `data` array. This was the array used in the sentiment and topic analysis.

4.8 Threads vs Multi-processes

Due to the nature of the project, it was necessary to implement a concurrency solution to manage the various messages and events to be analysed at the same time. Naturally, it was necessary to choose between programming with threads or with multi-processes. This subchapter describes each of the methodologies and how their solutions were implemented, the results of each are also discussed and a comparison between both.

Threads and multi-processes are two ways to achieve concurrency in computer programs, that is the ability of a program to execute multiple tasks simultaneously.

A thread is a unit of execution within a process that can run concurrently with other threads in the same process, which means that is managed by a scheduler. On the other hand,

[22] https://github.com/xenova/chat-downloader
[23] https://developers.google.com/youtube/v3?hl=pt-br

multiprocessing is a mode of operation in which multiple processes simultaneously run two or more different portions of the same program. A process can have one or many threads.

Threads are generally more efficient than processes because they have less overhead. This is because threads share the same memory space and resources as their parent process, while processes have their own memory space and resources. Additionally, threads can share data more easily than processes, which can further improve efficiency.

4.8.1 Threads

A thread is a small sequence of instructions that were programmed and shares the same memory space as other threads within the same process. This implies that threads have the ability to communicate with each other by exchanging data via shared variables. However, this implies that threads must be carefully synchronized to avoid problems or issues related to concurrency (Lee, 2006).

During the execution of a thread, the CPU creates the illusion that is performing multiple computation simultaneously. This is possible because it dedicates a small amount of time to each computation and maintains execution context for each computation.

Threads are particularly useful in scenarios of lengthy processing, for example when an application is calculating cannot process any more messages and as a result, the display cannot be updated. Is also useful for doing background processing and for doing I/O work.

4.8.2 Multi-processes

Multiprocessing if the ability of a system to execute more than one processor at the same time. In such a system, applications are divided into smaller, independent routines. The operating system allocates these, improving the performance of the system (Dijkstra et al., 1979).

A process is a heavier unit of execution that has its own memory space. This implies that processes are unable to directly share data and must rely on IPC mechanisms such as pipes, sockets, or shared memory for information exchange. While this can simplify the avoidance of concurrency issues, it may also complicate communication between processes.
In general, the scenarios where multiprocessing is useful are the same as the threads. The choice between one or another will depend on each application and use case.

4.8.3 Developed Alternatives

Despite the main approaches being threads and multi-processes, within these there were also two alternatives. One in which each request was a thread/process, and the API requests were made within these sequentially and whenever the previous analyses ended. In the second alternative, in addition to the thread/process that manages the requests, there is an extra one within these to manage the corresponding API request and in this way to be simultaneously

obtaining new comments while the previous analyses are taking place and theoretically improving the performance of the developed tool. The results are in the following sub-chapters.

Regardless of the use of threads or multi-processes, the creation and configuration of the consumer and producer is the same.

For the creation of the "master_consumer", the `KafkaConsumer` method from the Kafka repository was used. The parameters it receives are:

- "topic": The Kafka Topic that the consumer will be "listening" to, "master_topic" in this case.
- "bootstrap_servers": Where the IP of the ISEP server is indicated.
- "security_protocol": Indicates the chosen security protocol, "PLAINTEXT" was the chosen here.
- "auto_offset_reset": Indicates which messages to show first. To show the more recent messages first, the "latest" option was the chosen.
- "group_id": The id of the Kafka Group in which the consumer is inserted, "master_consumer_group", in this case.

For the creation of the producer, the `KafkaProducer` method was used, which receives the following parameters:

- "bootstrap_servers": Where the IP of the ISEP server is indicated.
- "value_serializer": Parameter that allows to indicates a method to serialize the Kafka Messages.
- "security_protocol": Indicates the chosen security protocol, "PLAINTEXT" was the chosen here.

4.8.3.1 Approaches using threads

The master consumer is waiting to receive messages from MOG with analysis requests. When it receives one, it checks whether the message is a "start" or an "end", as seen in 4.6.2.1.

In the case that the message is to start the analysis, an `add_request` method is called, in this method a thread is created for the request, and this is added to the array of requests that are being executed. This thread executes the `process_request` method. There is a loop that checks if the request is still in the array, that is, it will always be running as long as it is present in the request array. Within the loop is the verification of the social network to analyse (reddit, YouTube or internal chat). In the case of chat, the comments to be analysed are in the request message and so they are immediately added to the `data` array, which will undergo inference by the models. If the network to be analysed is Reddit, the `get_reddit_data` method is executed. In this method, all comments from the desired URL are obtained through PRAW (as

described in 4.7.1). If the network is YouTube, the `get_youtube_data` method is executed, where comments are obtained with the chat_downloader. It is verified which analysis to perform, and a thread is created for each analysis, that is, if it is to perform sentiment analysis and topic analysis, two threads are created, in the case of being only one of these analyses, only one thread is created. Depending on the type of analysis there is a corresponding method that is called in the corresponding process, `start_SA_analysis` for sentiment analysis and `start_TA_analysis` for topic analysis. These methods are very similar, with the difference that one calls the sentiment analysis model to perform the inference and the other the topic analysis model. When the results are returned by the models, a message is created by the `create_sentiment_message_json` or `create_topic_message_json` method, which will be sent to MOG as described in 4.6.2.2, and are immediately sent by the `send_message` method, which uses a Kafka method to perform this sending. Subsequently, it is checked again if it was an analysis of chat comments, and if that is the case the request is removed from the array of requests that are being executed and the loop does not proceed. This loop has a minimum time of one minute, that is, if the analyses take longer than that to be executed, the loop automatically advances to the next iteration, if the analyses are fast and less than this time, the remainder is waited until it makes up sixty seconds and only then proceeds to the new iteration.

In the case of the "end" message, the request is removed from the array of processes that are running and it is terminated.

4.8.3.2 Approaches using multi-processes

Obtain data in sequence

The master_consumer waits for a new message from MOG with analysis requests. When that message arrives, a method called `process_message` is executed. The first step in the method is to verify is the messages is to start an analysis or to end one.

In the case of the "end" message, the request is removed from the array of processes that are running and it is terminated. If that process is running an analysis, first the analysis is finished and then the process is terminated.

In the case of the "start" message, there is a second check to confirm whether an equal event is already being processed or not. If it is a new event is added to the array of requests and a process for it is created. This process will execute a method called `process_request`. In here there is a if condition to determine if the analysis will be from a YouTube livestream, Reddit post or comments from the internal chat. In case of YouTube or Reddit, a different method to get the comments from those platforms is called, `get_youtube_data` or `get_reddit_data`, respectively. Those connect to the respective API and get the comments from the URL provide in the "channel" parameter from the request message. After obtain the comments, and if there are any, the type of analysis is confirmed, this is whether it is necessary to do sentiment analysis

and topic analysis or just one of them. For each one is created a process that will oversee that analysis. Depending on the type of analysis there is a corresponding method that is called in the corresponding process, `start_SA_analysis` for sentiment analysis and `start_TA_analysis` for topic analysis. These methods are very similar, with the difference that one calls the sentiment analysis model to perform the inference and the other the topic analysis model. When the results are returned by the models, a message is created by the `create_sentiment_message_json` or `create_topic_message_json` method, which will be sent to MOG as described in 4.4.2.2 and are immediately sent by the `send_message` method, which uses a Kafka method to perform this sending. After this, if the comments analysed were from chat, the request is automatically removed from the array of requests. Them the time from the last comments analysed is updated, this is used to guarantee that there are no comments to be analysed more than once.

Obtain data simultaneously

The master_consumer is waiting to receive messages from MOG with analysis requests. When a message is received, a method called `process_message` is executed. In this method, it is checked whether the message is a “start” or an “end”, as seen in 4.6.2.1.

In the case of being a start processing message, there is a second check to confirm whether an equal event is already being processed or not. This ensures that there are no duplicates, even if messages from MOG arrive, making requests for repeated events (determined by the id plus the URL). In the case of being a new event, it is added to the array of requests that are being analysed and a process is created for it. This process will execute a method called `process_request`. In the `process_request` method, there is a first check to understand if the request is to analyse chat messages or if it is to analyse comments from one of the social networks. If the request is to analyse messages from the platform’s internal chat, the `process_analysis` method is called directly (it is in this method that the machine learning models are called) and it is immediately removed from the array of requests that are being analysed. In the case of being to analyse comments from social networks, a new process is created that manages the `get_data` method and in a loop, while the request process is being executed, the `process_analysis` method will be called to proceed with the inference of the machine learning models. The data (comments) that will be analysed are in the data array and are data that are always being added in parallel by the `get_data` method (which is being managed by a separate process), whenever an analysis ends, the analysed comments are removed from this array and a new analysis is started with the new comments added, meanwhile. It is in the `get_data` method, that the process described in 4.7 is executed. In the `process_analysis` method, a process is created for each analysis to be performed, for example if the request message indicates that it is to perform sentiment analysis and topic analysis then two processes are created, if it only indicates one of these, only one process is created. Depending on the type of analysis there is a corresponding method that is called in the corresponding process, `start_SA_analysis` for sentiment analysis and

`start_TA_analysis` for topic analysis. These methods are very similar, with the difference that one calls the sentiment analysis model to perform the inference and the other the topic analysis model. When the results are returned by the models, a message is created by the `create_sentiment_message_json` or `create_topic_message_json` method, which will be sent to MOG as described in 4.4.2.2 and are immediately sent by the `send_message` method, which uses a Kafka method to perform this sending.

In the case of the "end" message, the request is removed from the array of processes that are running and it is terminated.

4.8.4 Results

In this sub-chapter the results of the three developed approaches for running this project. It should be noted that the different types and formats of Kafka messages have no influence on performance and all these tests were performed using the latest version of these messages 4.6.

All tests were performed using the same data set for each social network. 2000 Reddit comments were used, which were used in the three different tests, 2000 YouTube comments were used, which were used in the three tests and 500 chat comments were used, which as in the other cases, were used in the three tests. The preference for using the same comments was because, that way, it showed the real performance of the solution without the size of the data set or each comment interfering with the tests.

4.8.4.1 Approaches using threads

In this approach, all iterations of each request are sequential, that is, an event with a certain channel and corresponding social network. But the analysis of another social network for the same event is done in parallel and it is only here that the threads come in.

YouTube

With the timestamp values obtain during the sentiment analysis test, was possible to construct Figure 7. In this are represented (in blue) the total time in seconds that each iteration ran, that is, the total time of each analysis, including the time to obtain the comments. Also represented is the time of each inference (in red), that is, only the execution time of the sentiment analysis model. The green line represents the number of comments, which is always constant.

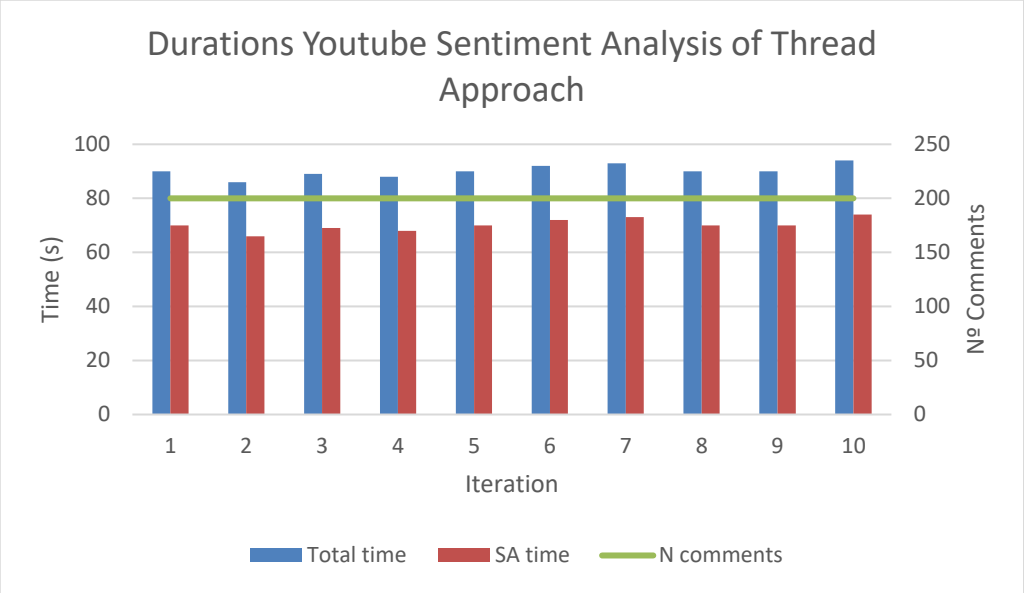

Figure 7 - Durations Youtube Sentiment Analysis of Thread Approach

From Figure 7 it is possible to verify that the execution time is very similar in this approach. This occurs because the number of comments is the same and the small differences can derive from the size of each comment within the set analysed in that iteration and from other analyses that would be taking place at the same time. The average run time was 90.2 seconds in total and 70.2 seconds for the inference.

With the values obtained in the topic analysis test, was possible to make Figure 8. In this are represented in blue the total time in seconds that each iteration ran, in red the time of each inference and the green line represents the number of comments, which is always 200.

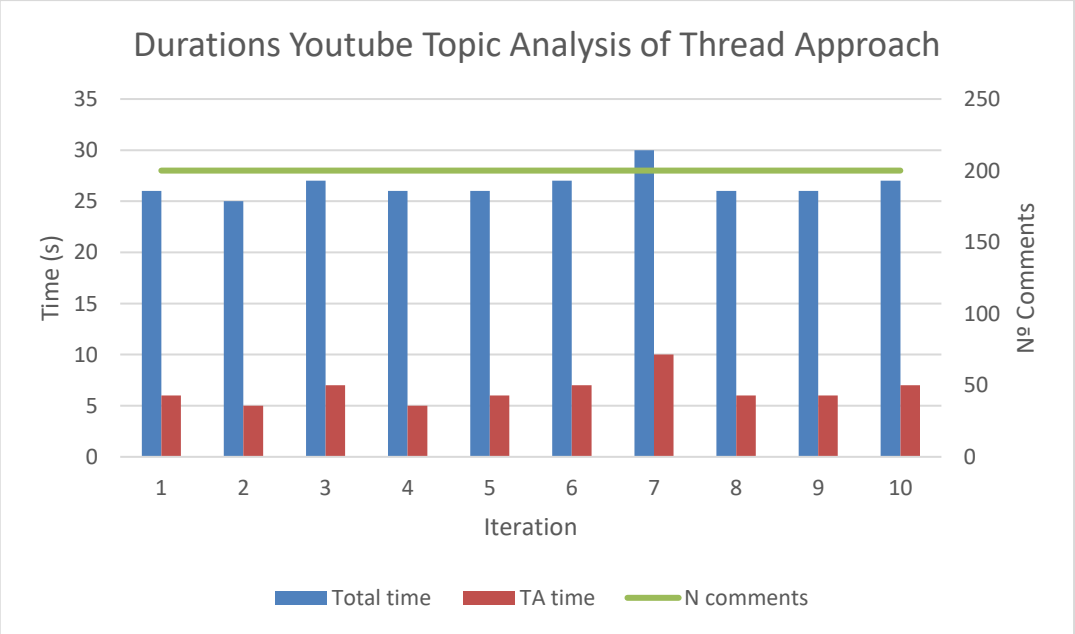

Figure 8 - Durations YouTube Topic Analysis of Thread Approach

It is possible to conclude, as in the sentiment analysis that the values are similar in all iterations, although in this case there is a greater difference. There is also a more pronounced difference between the acquisition time and the inference time, but this occurs because the inference time is greatly reduced. On average it was 6.5 seconds, and the total time was 26.6 seconds.

The Figure 9 compares the times of the inferences and is possible to verify that the sentiment analysis is much slower than the topic analysis.

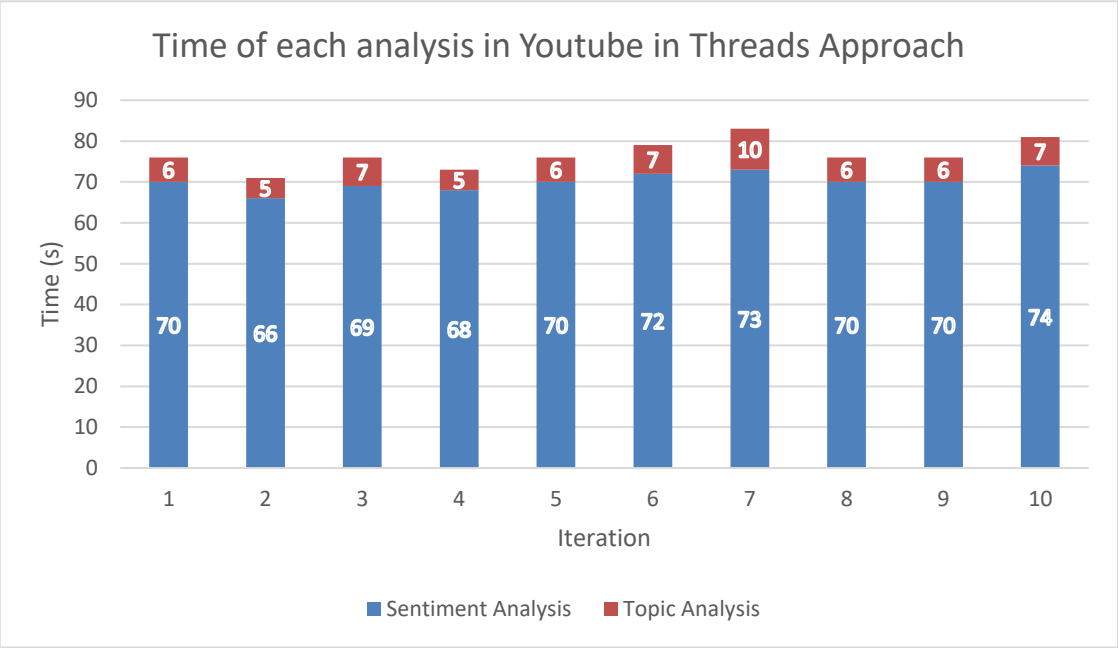

Figure 9 – Time of each analysis in YouTube in Threads Approach

Reddit

The Figure 10 is the representation of the values of the sentiment analysis test. Like in the YouTube sentiment analysis is possible to see a constant amount of time through each iteration and the majority of the total time is spent doing the inference. The average of the total time is 91. seconds and the average of the inference is 71.2 seconds, which means that the inference is 77.98% of the total time.

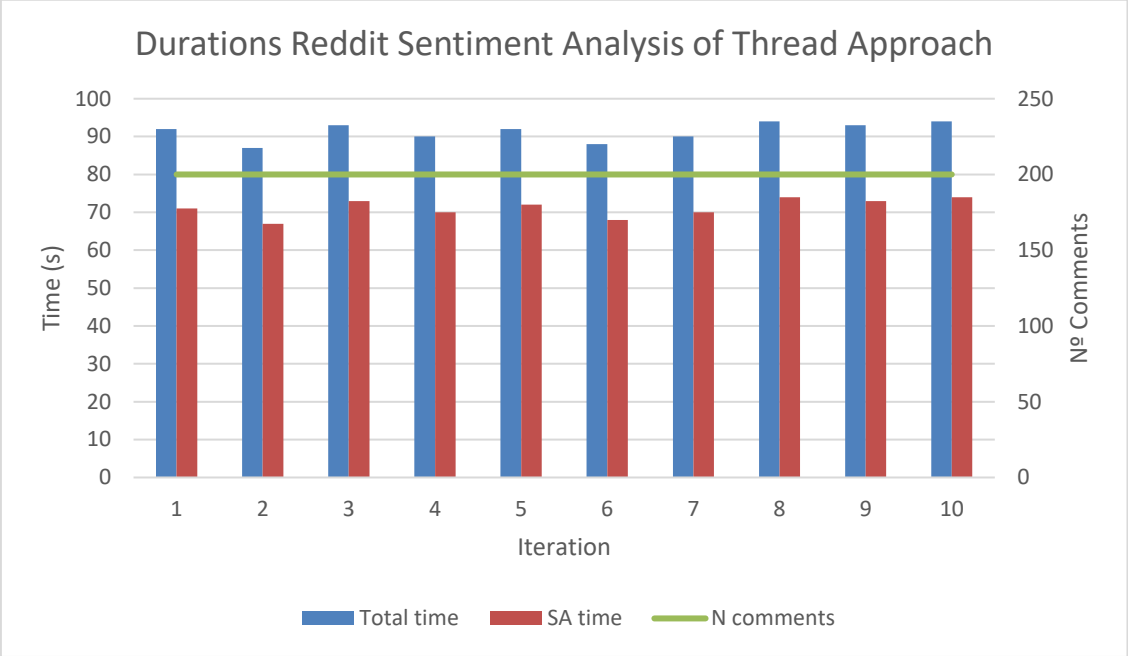

Figure 10 - Durations Reddit Sentiment Analysis of Thread Approach

Analysing Figure 11, is possible to see that in this analysis there were much more variations in the time it took them to execute. The shortest were the second, third, sixth and seventh iteration, with 26 seconds of total time and 6 seconds in the inference. The longest was the fifth iteration, it took 33 seconds in total and 13 seconds to infer.

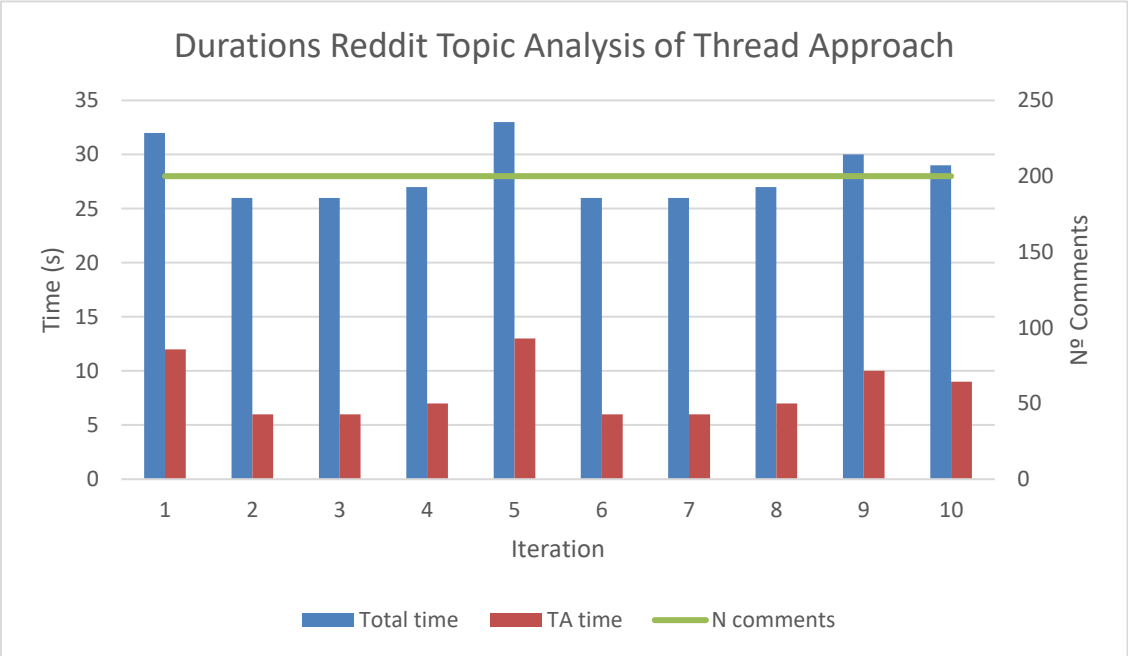

Figure 11 - Durations Reddit Topic Analysis of Thread Approach

In the Figure 12, are represented the seconds that each analysis took. Once again, the sentiment analysis took much longer than the topic analysis. The sentiment analysis took, in average, 71.2 seconds, while the topics analysis only took 8.2 seconds.

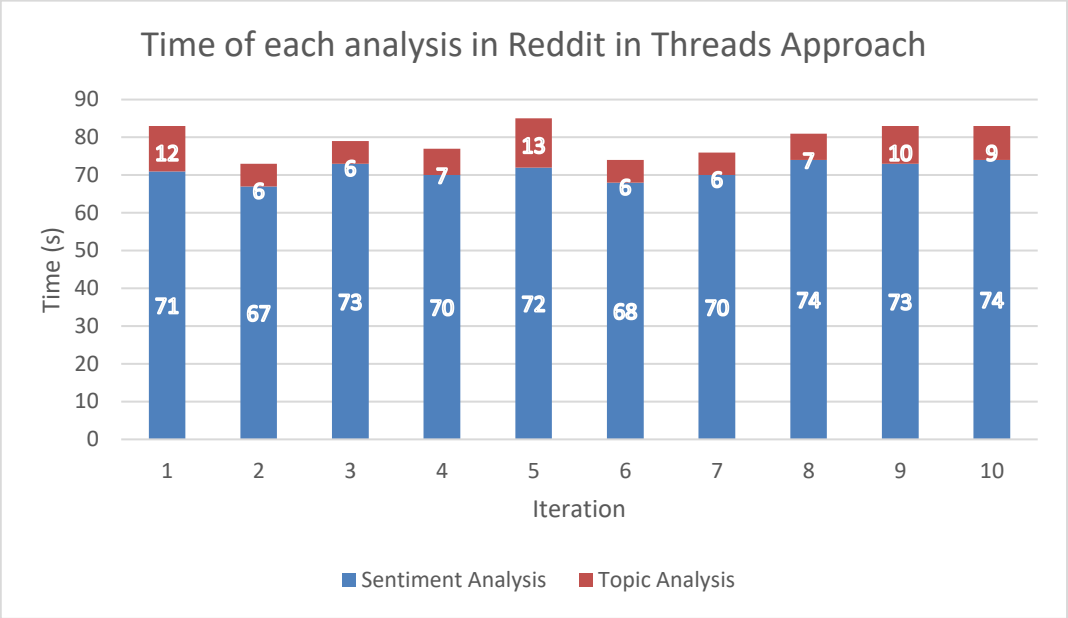

Figure 12 - Time of each analysis in Reddit in Thread Approach

Chat

The iterations with comments from chat are different, because is necessary to have a request message for each iteration and there aren't timestamp for requests to the APIs because all the comments come in the Kafka Message.

The Figure 13 show that almost all the time in the execution was to the inference, which in theory should be the case because there's no need to get data from social media.

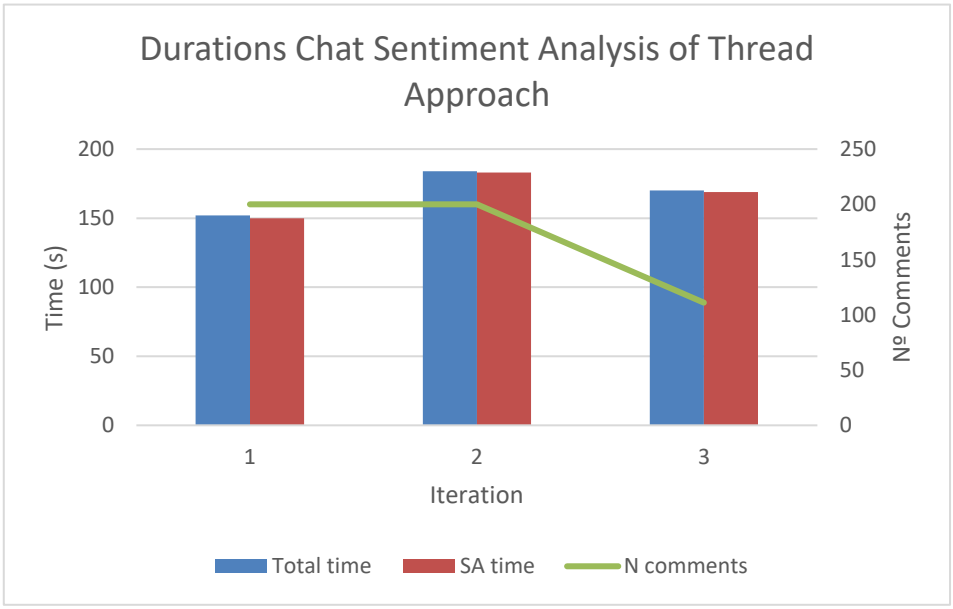

Figure 13 - Durations Chat Sentiment Analysis of Thread Approach

In the chat analysis using threads, each time a request entered the system the total time of execution increased, and the majority of the total run time continue to be for the inference, as it can be seen in Figure 14.

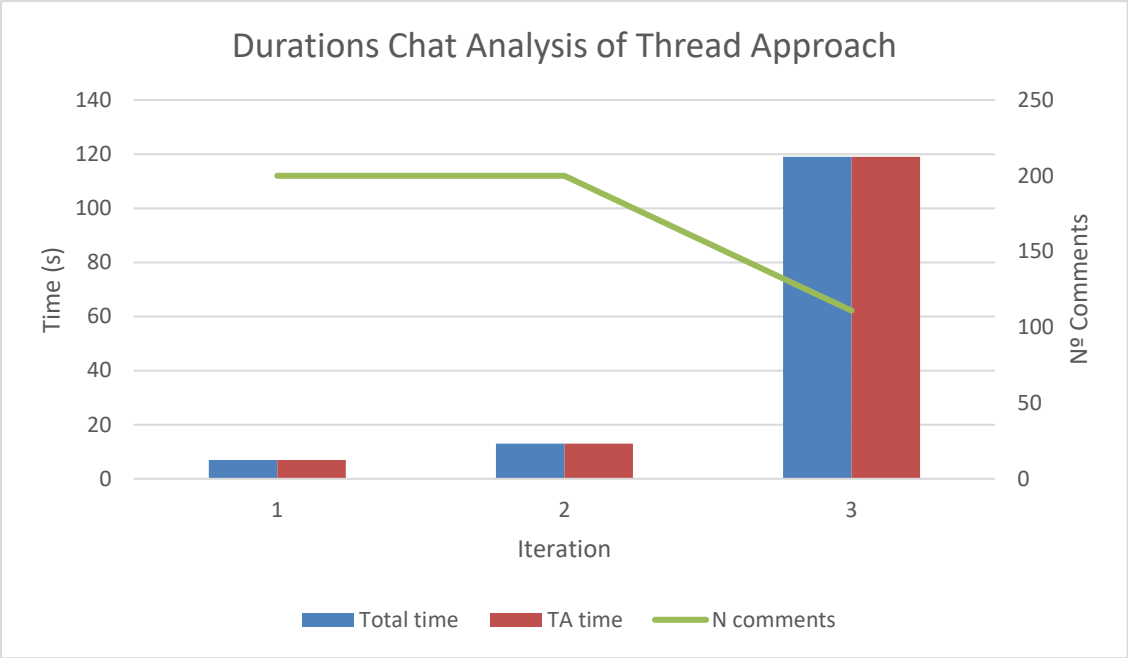

Figure 14 - Durations Chat Analysis of Thread Approach

When comparing the time, it took to running the sentiment analysis and the topic analysis, the sentiment analysis was slower than the topic analysis. In this case way longer, as can be seen in Figure 15. In the first iteration it took 21.43 times longer to run the sentiment analysis, in the second iteration 14.08 times longer but in the last iteration it only took 1.42 times longer.

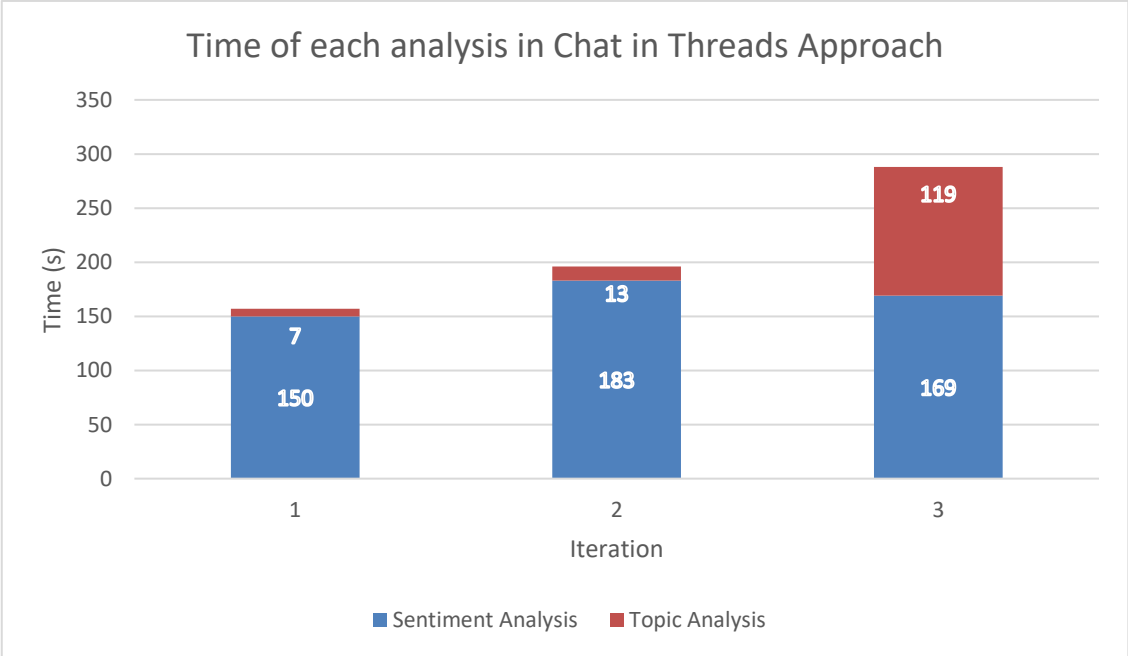

Figure 15 - Time of each analysis in Chat in Threads Approach

4.8.4.2 Approaches using multi-processes

Sequential

YouTube

Figure 16 was constructed based on the values of the sentiment analysis test. In this are represented (in blue) the total time in seconds that each iteration ran, that is, the total time of each analysis, including the time to obtain the comments. Also represented is the time of each inference (in red), that is, only the execution time of the sentiment analysis model. The green line represents the number of comments, which is always constant.

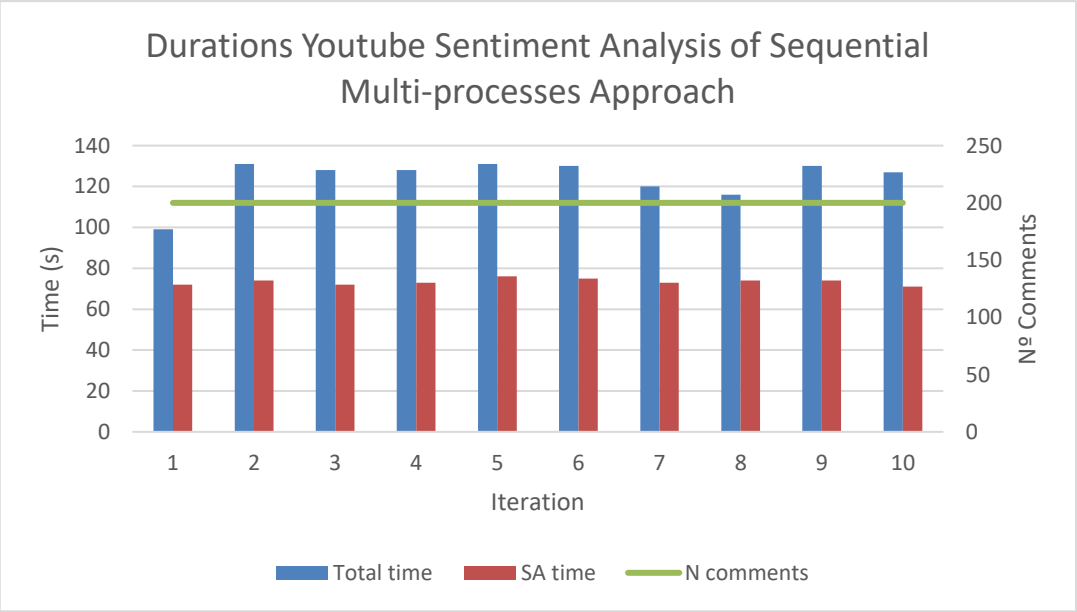

Figure 16 - Durations Youtube Sentiment Analysis of Sequential Multi-processes Approach

The green line is constant because the number of comments is always 200. Is perceptible that the time to run each iteration is consistent throughout all execution, apart from the first one, which was the fastest. The time that it takes to make the inference corresponds to about 59.19% on average of the total time.

Figure 17, shows some variation in the number of seconds it takes to execute each iteration. The first was the fastest overall, it took 49 seconds, of which 15 in inference, but the fastest inference occurred in iteration 7. On the other hand, iteration 9 was the longest, having been running for 90 seconds.

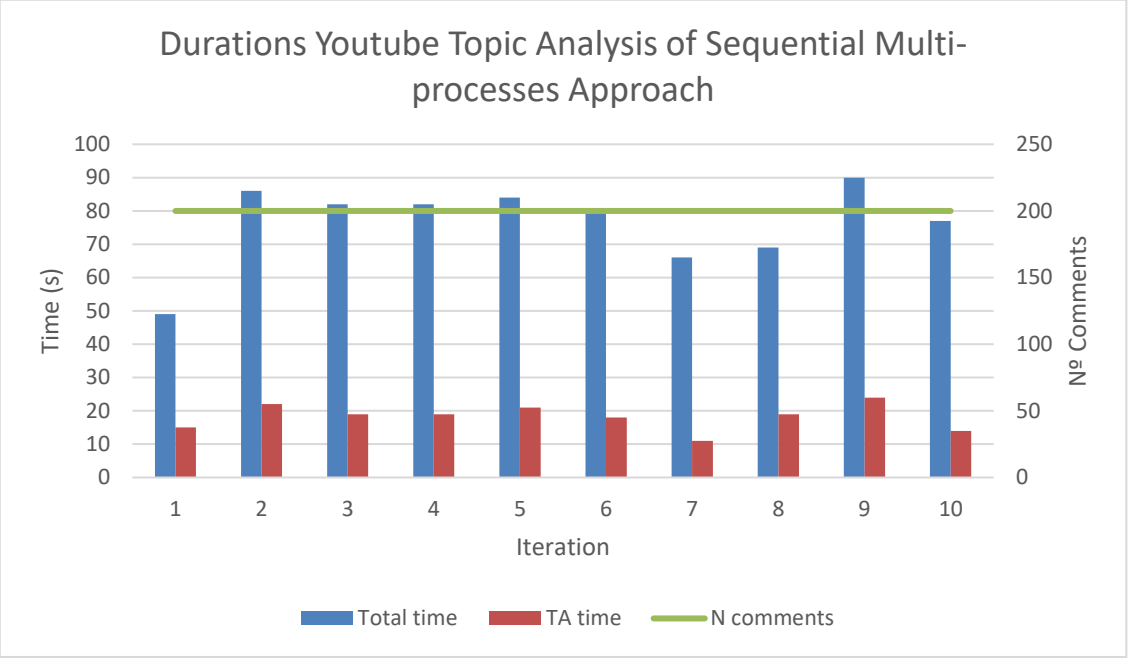

Figure 17 - Durations Youtube Topic Analysis of Sequential Multi-processes Approach

Comparing both analysis is possible to see that the sentiment analysis continues to be the one that takes longer, no matter the approach. In this case it took 73.4 seconds against 18.2 seconds on average.

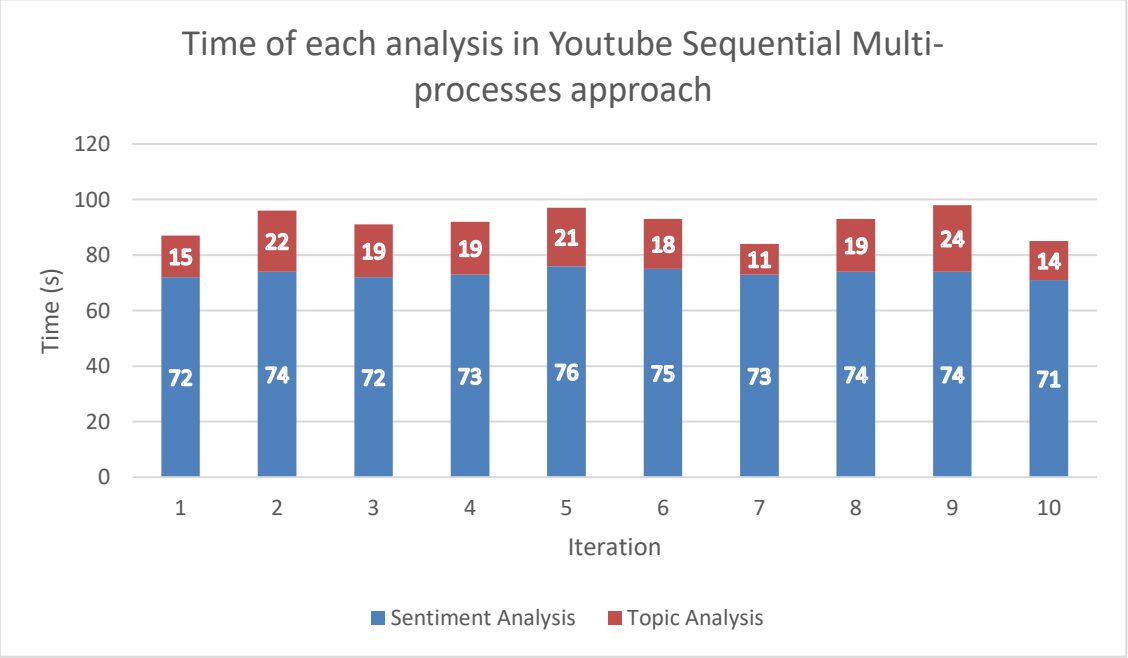

Figure 18 - Time of each analysis in YouTube Sequential Multi-processes approach

Reddit

It took on average 73.5 seconds to run the inference of the sentiment analysis model and 121.2 seconds on total, which means that this approach took more than two minutes to run the sentiment analysis of comments from reddit. This can be seen in Figure 19.

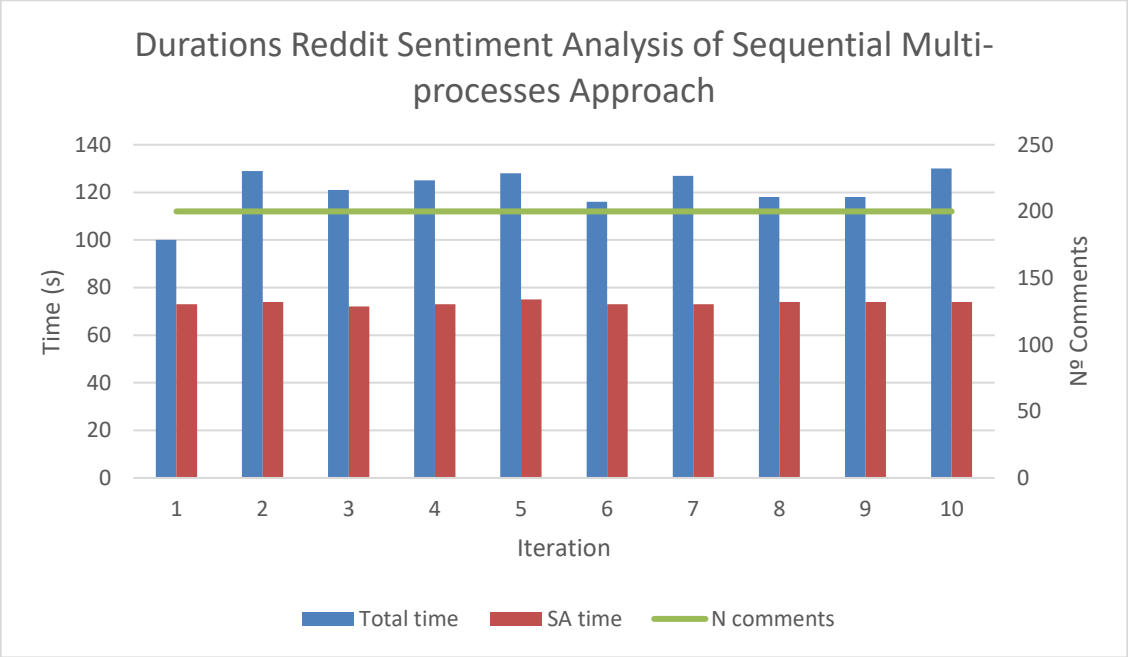

Figure 19 - Durations Reddit Sentiment Analysis of Sequential Multi-processes Approach

From the topic analysis tests, is possible to create the Figure 20. Here from iteration number two until the sixth, the total time was decreasing, while the inference time was fluctuating, but on the seventh iteration the total time spike, hitting the maximum amount of this test, with 90 seconds.

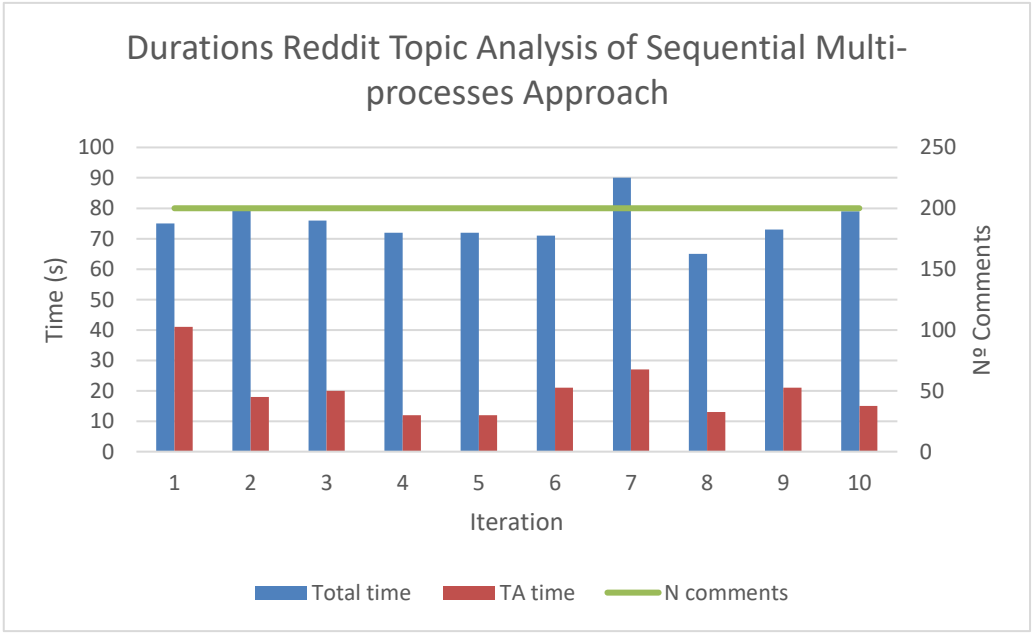

Figure 20 - Durations Reddit Topic Analysis of Sequential Multi-Processes Approach

With the values from Figure 19 and Figure 20 is possible to create Figure 21. It allows to compare the times that both inferences took to execute. Once again, sentiment analysis took longer than the topic analysis.

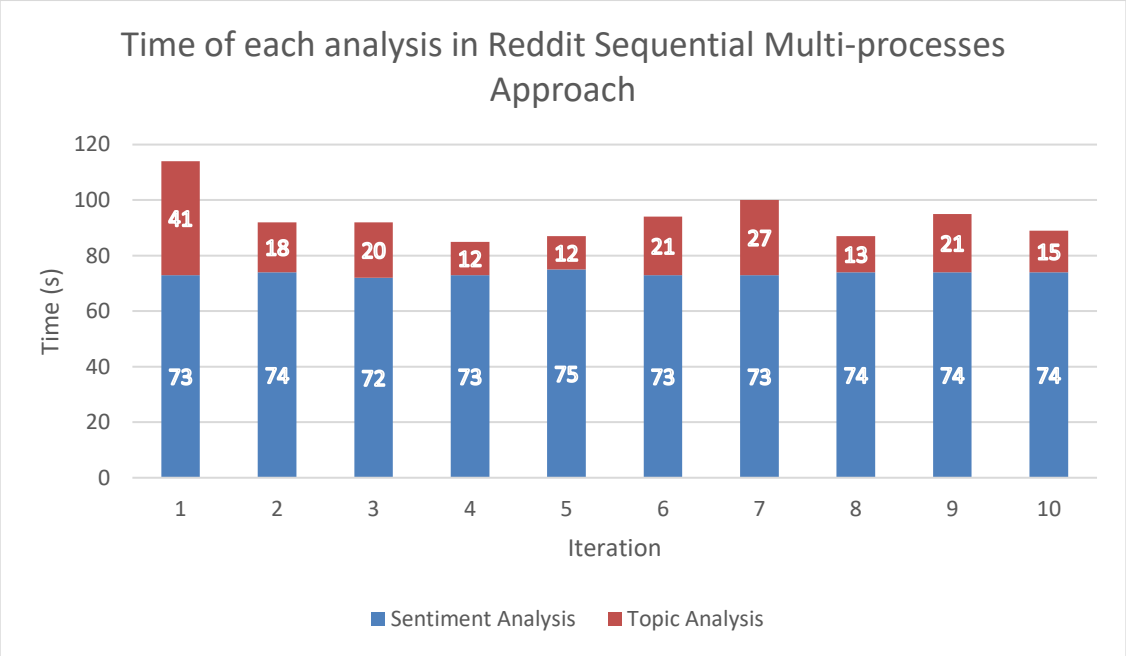

Figure 21 - Time of each analysis in Reddit Sequential Multi-Processes Approach

Chat

The iterations with comments from chat are different, because is necessary to have a request message for each iteration and there aren't timestamp for requests to the APIs because all the comments come in the Kafka Message.

The Figure 22 show that almost all the time in the execution was to the inference, which in theory should be the case because there's no need to get data from social media, but in the third iteration that's not the case. There's no explanation for that in this test and others that were conducted.

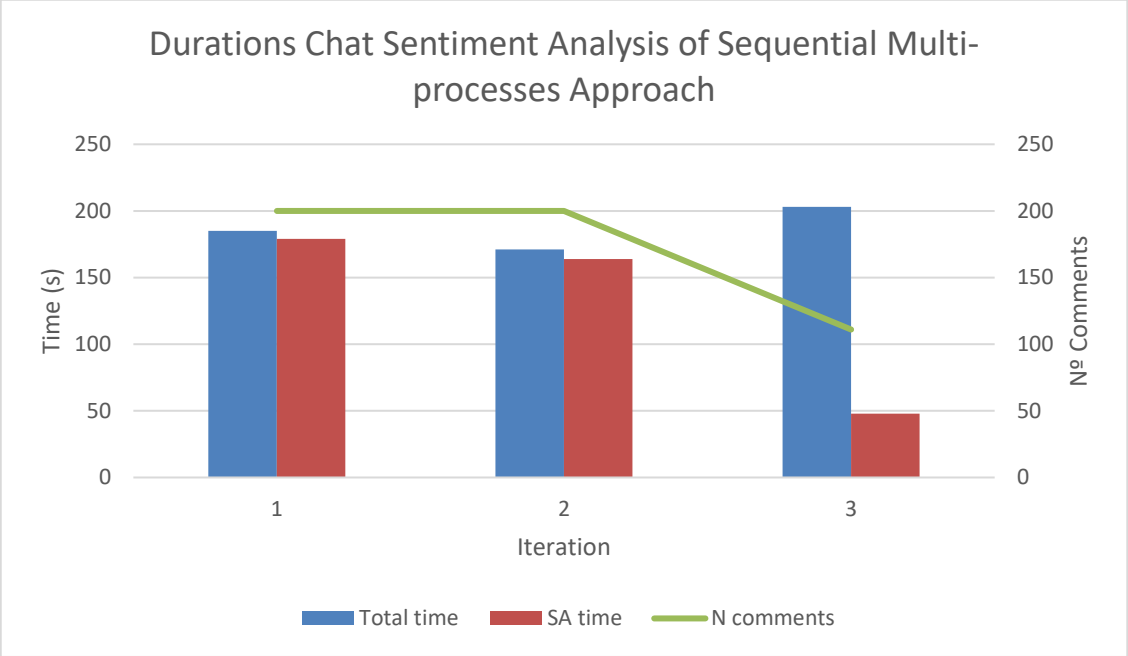

Figure 22 - Durations Chat Sentiment Analysis of Sequential Multi-processes Approach

In the topic analysis using sequential multi-processes approach, each time a request entered the system the total time of execution increased, on the other hand the time of inference decreased ().

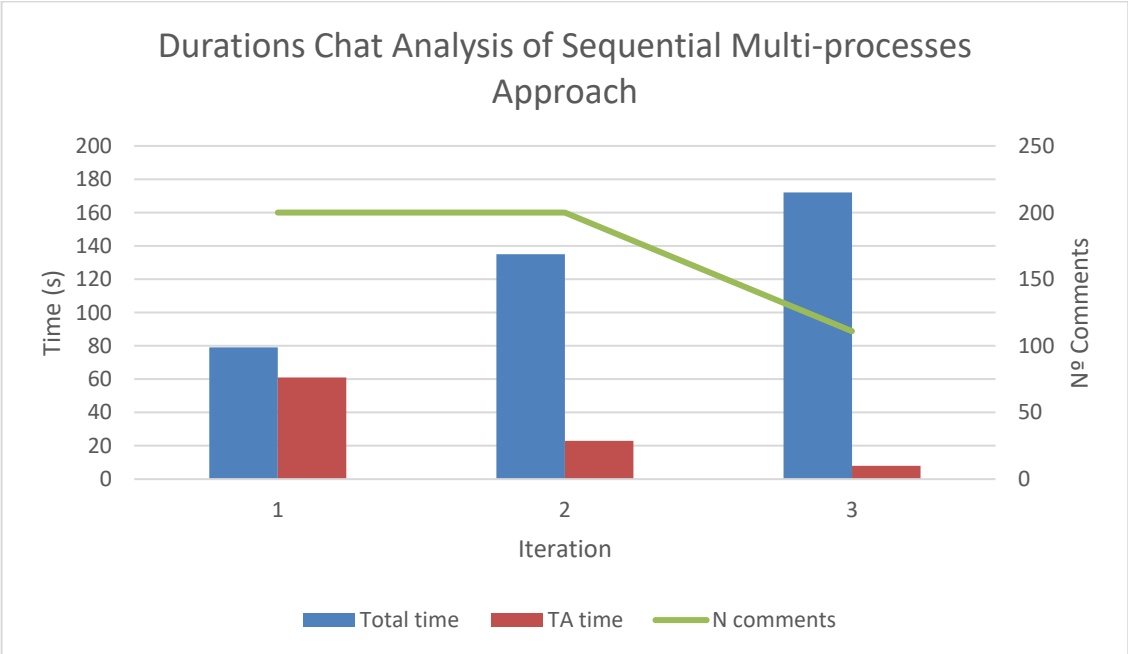

Figure 23 - Durations Chat Analysis of Sequential Multi-processes Approach

When comparing the time, it took to running the sentiment analysis and the topic analysis, like in every previous approach, the sentiment analysis took longer than the topic analysis. In this case way longer, as can be seen in Figure 24. In the first iteration it took 3 times longer to run

the sentiment analysis, in the second iteration 7.13 times longer and lastly in the third iteration it took 6 times longer.

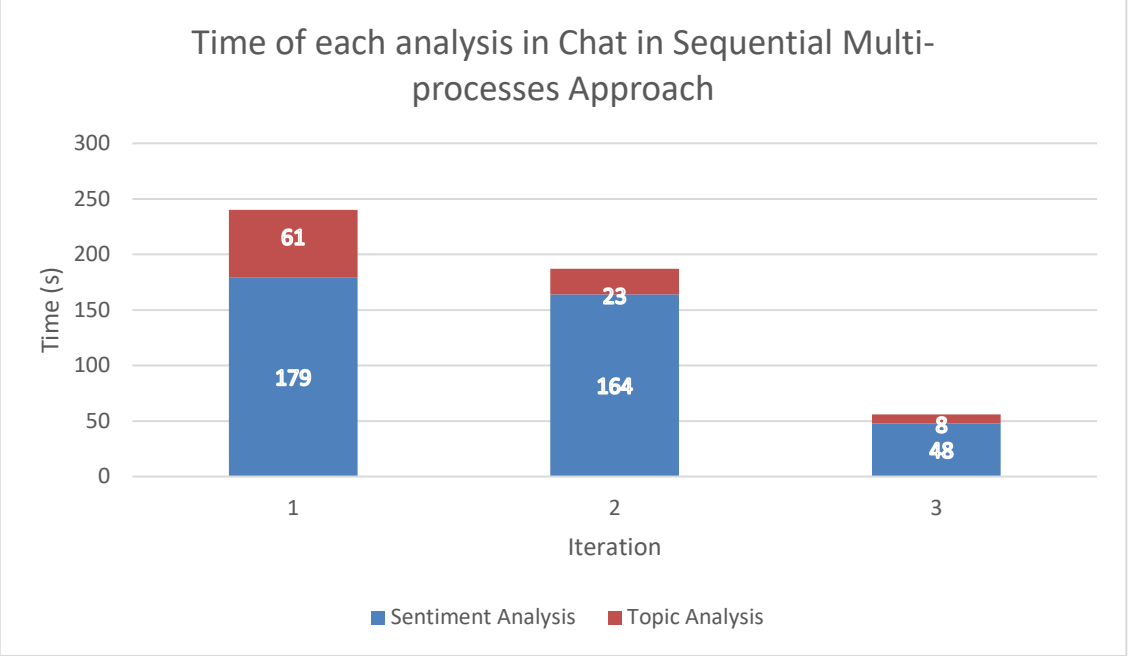

Figure 24 - Time of each analysis in Chat in Sequential Multi-processes Approach

Parallel

Here, all the analyses are being executed in parallel, as well as the obtaining of the comments. This is the change from the sequential approach.

YouTube

Looking at Figure 25, the total execution time was increasing while the sentiment analysis time remained relatively constant. But unlike the previous approaches, it is not possible to obtain a correlation between these two values, because while an analysis is being executed, the program is already reading more comments, that is, in this case the most important values are the final values from the beginning of the first iteration to the end of the last one and the values of each inference.

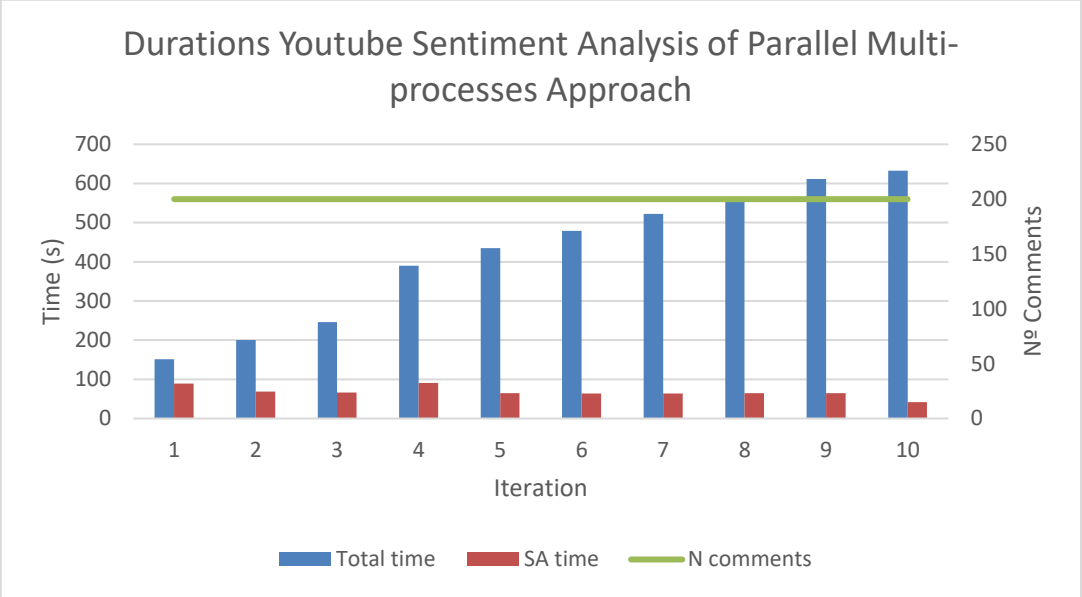

Figure 25 - Durations Youtube Sentiment Analysis of Parallel Multi-processes Approach

In the case of the topic analysis (Figure 26), the first three iterations were faster than the others. This could happen because the server started to get multiple processes running at the same time, which decreased the performance. Here it is not possible to make a direct correlation between the total time and the inference time.

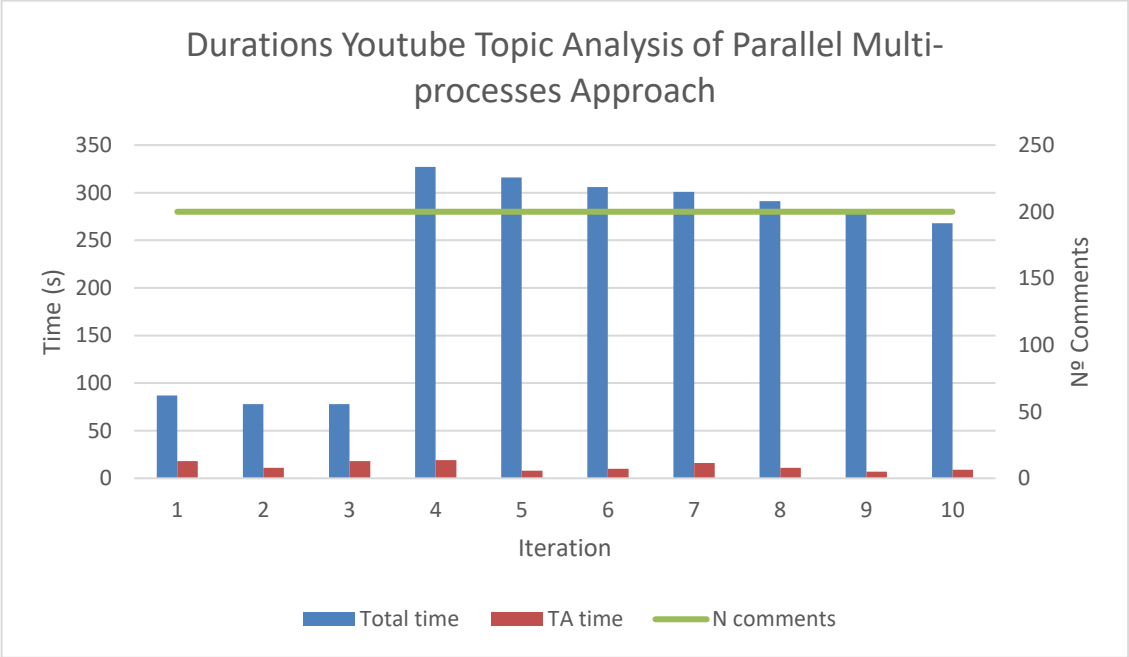

Figure 26 - Durations Youtube Topic Analysis of Parallel Multi-processes Approach

The sentiment analysis continues to be the slowest, here it took 68 seconds in average and the topic analysis took 12.7 seconds, as can be seen in Figure 27.

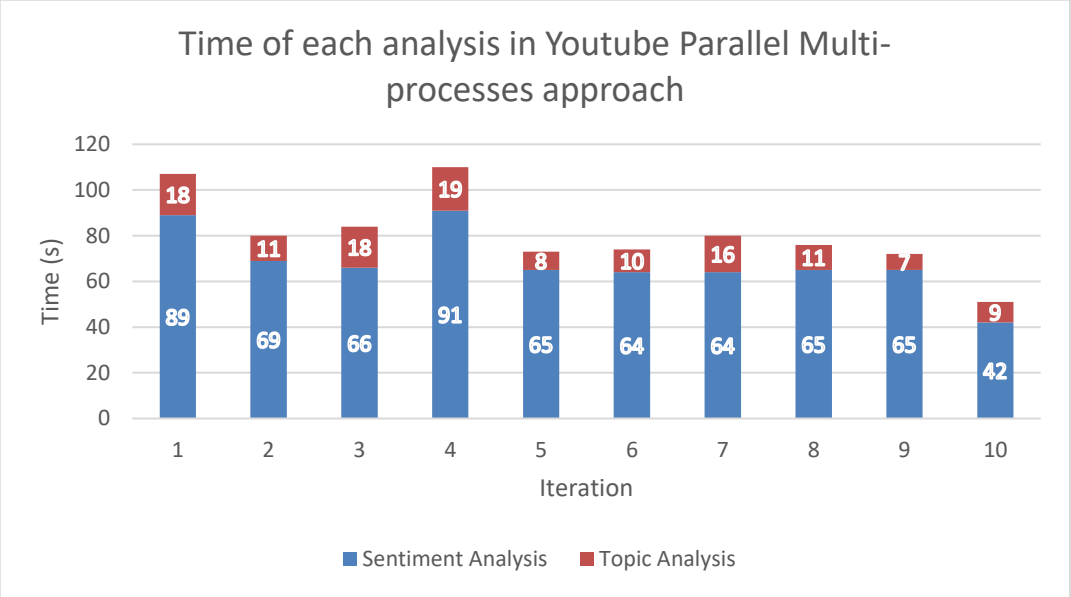

Figure 27 - Time of each analysis in YouTube Parallel Multi-processes approach

Reddit

The same phenomenon that happened in the sentiment analysis of YouTube comments, happened here. In each iteration the total time increased, while the inference time stays almost the same, as can be seen in Figure 28.

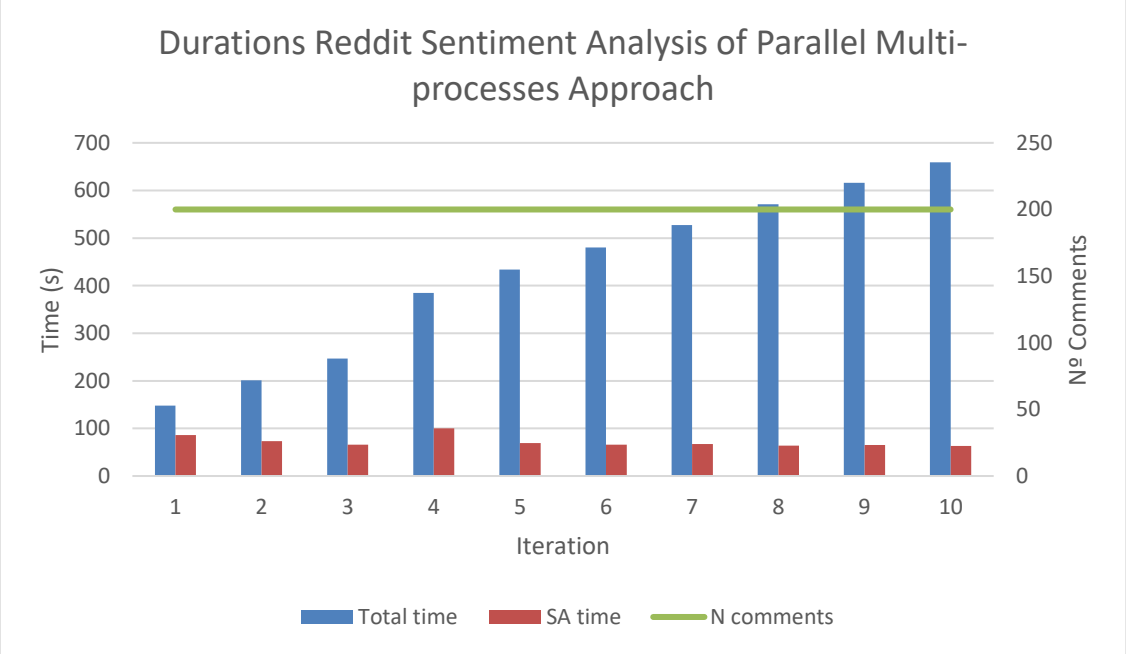

Figure 28 - Durations Reddit Sentiment Analysis of Parallel Multi-processes Approach

In the case of the topic analysis (Figure 29), and just like in YouTube, the first three iterations were faster than the others. This could happen because the server started to get multiple processes running at the same time, which decreased the performance.

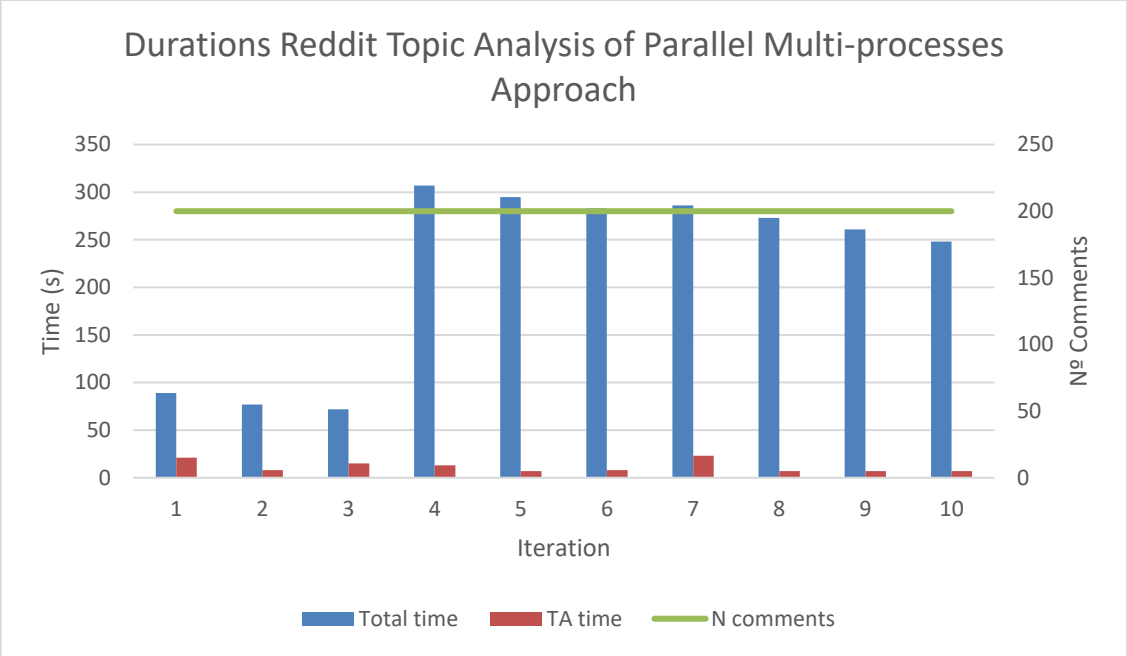

Figure 29 - Durations Reddit Topic Analysis of Parallel Multi-processes Approach

The sentiment analysis continues to be the slowest, here it took 71.9 seconds in average and the topic analysis took 11.6 seconds. The sentiment analysis was slower than in YouTube, but the topic analysis was faster. This comparison is represented in Figure 30.

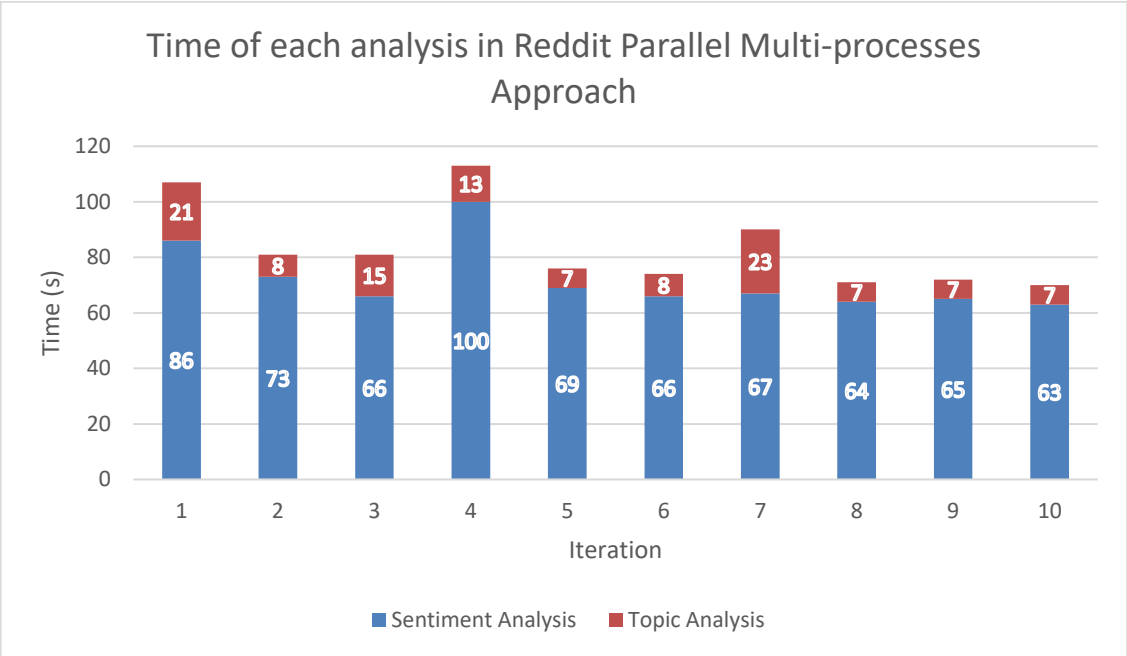

Figure 30 - Time of each analysis in Reddit Parallel Multi-processes Approach

Chat

The iterations with comments from chat are different, because is necessary to have a request message for each iteration and there aren't timestamp for requests to the APIs because all the comments come in the Kafka Message.

The Figure 31 show that almost all the time in the execution was to the inference, but like in the previous approach, the third iteration was different.

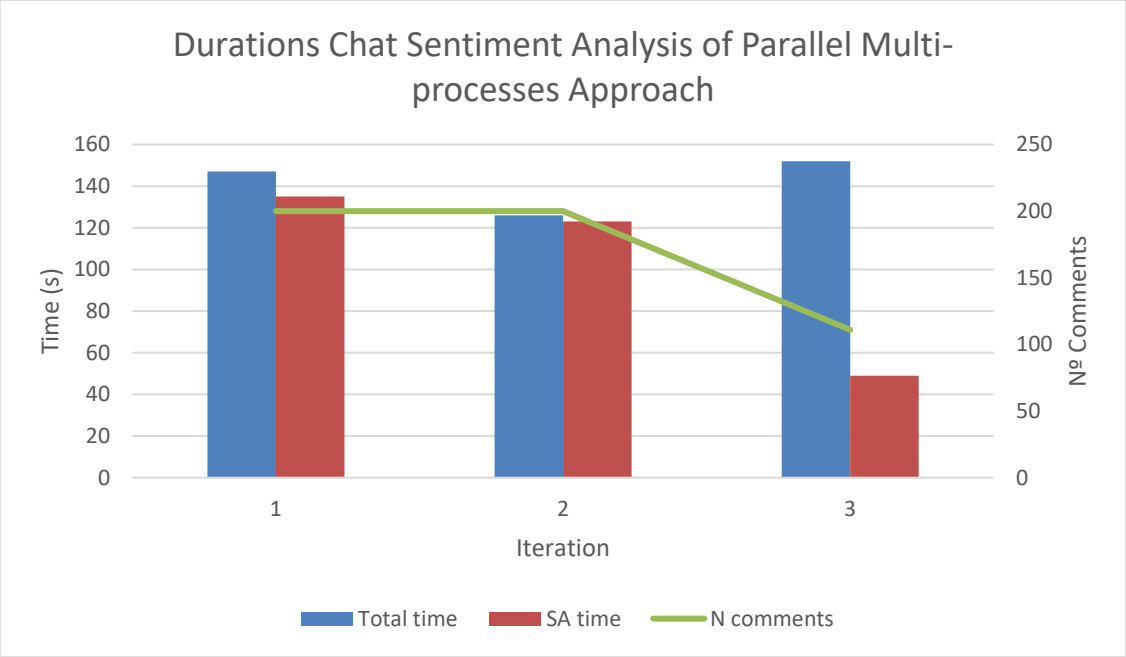

Figure 31 - Durations Chat Sentiment Analysis of Parallel Multi-processes Approach

In the topic analysis, each time a request entered the system the total time of execution increased. From the first to second iteration, the inference time also increased, but from there to the third decreased (Figure 32).

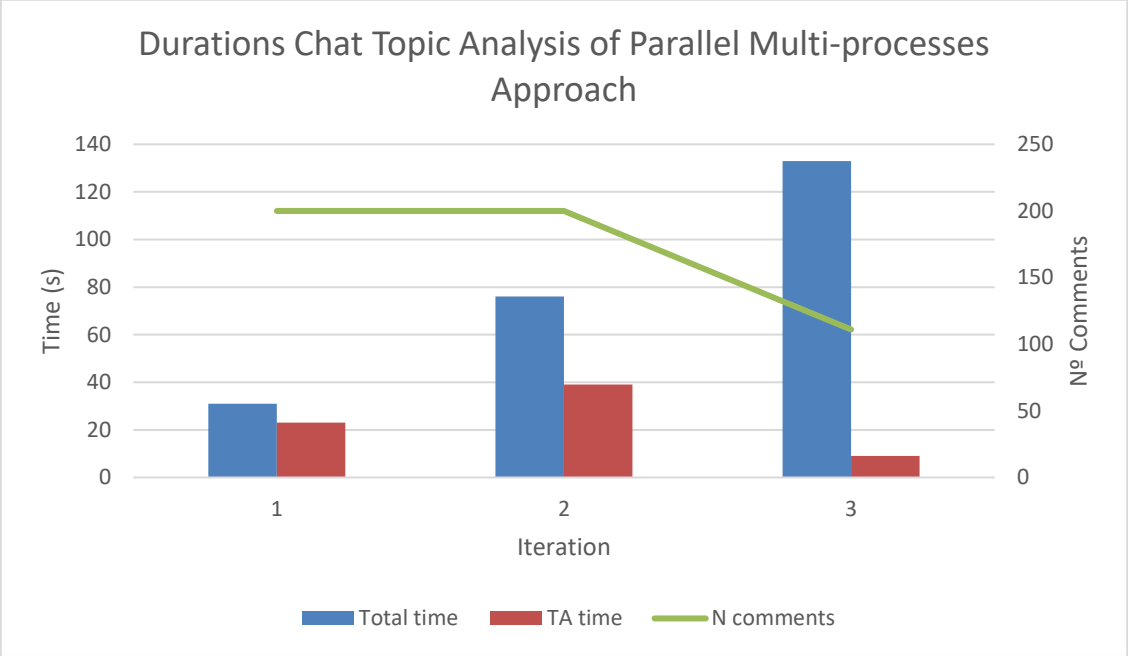

Figure 32 - Durations Chat Topic Analysis of Parallel Multi-processes Approach

It took to running the sentiment analysis 106.67 seconds on average and the topic analysis took 23.67 seconds. That's means that, once again, the topic analysis is the fastest, being 4.5 times faster than sentiment analysis.

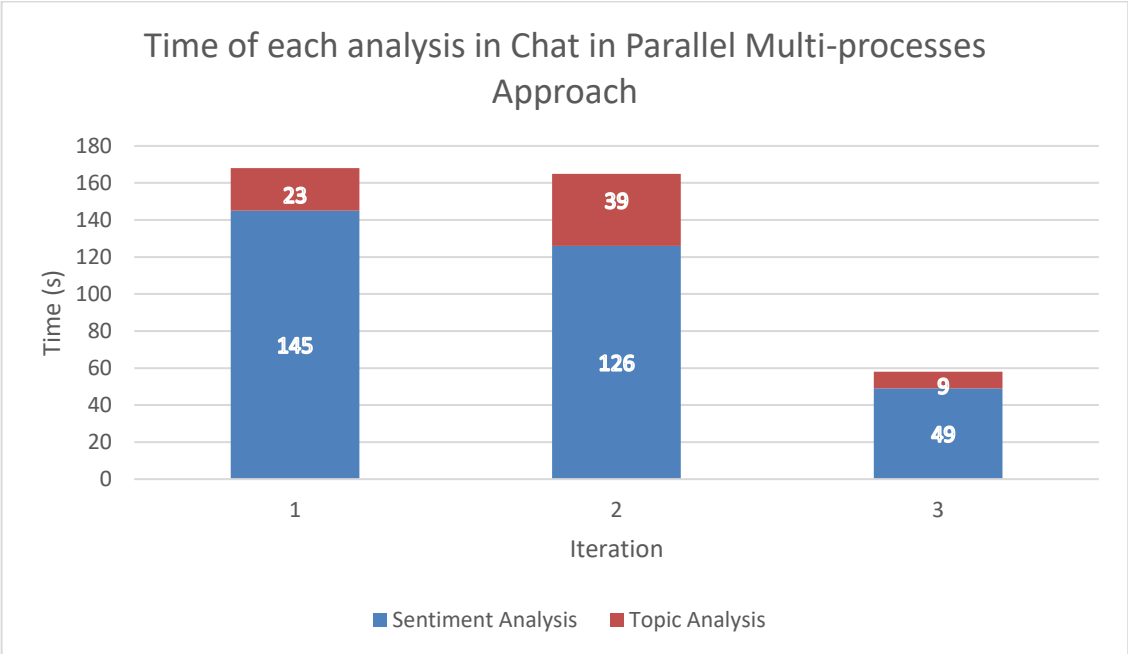

Figure 33 - Time of each analysis in Chat in Parallel Multi-processes Approach

4.8.5 Discussion of Results

In these tests, every social network had 2000 comments, separated by ten iterations of 200 comments each and the chat had 511 comments, separated by three iterations, two of 200 and one of 111. This way the dataset was the same for every approach and was possible to saw the difference between them. Overall, each approach analysed 4511 comments. The results, that is, the time in seconds that each approach took to analyse a social network is described in Table 2.

Table 2 - Comparison of the results of the tests

	Threads	**Multi-processes Sequential**	**Multi-processes Parallel**
YouTube	902	1252	943
Reddit	913	1224	865
Chat	239	244	203
Total	2054	2720	2011

The approach using threads in a sequential manner, took 2054 seconds (around 34 minutes) to analyse all of 4511 comments. The multi-process approach with getting the data in a sequential manner took the longest, 2720 seconds (around 45min) and the multi-process approach with parallel retrieved data, took 2011 seconds (around 33 minutes).

By the number of Table 2 and the results presented in 4.8.4, is possible to affirm that the approach using multi-processes with getting the data currently is the best one. And if the amount of data and events running at the same time, the difference will be even better (while the hardware do not reach its limit).

5 Changes to NLP models

To be able to connect the Kafka Structure with all its specifications and requirements, to the NLP models (sentiment analysis and topic analysis), it was necessary to modify certain parts of these. In this chapter, these changes are described and contextualized.

5.1 Sentiment Analysis Model

The create_dataloader method of the data.py file, had the parameters it receives modified. In the modified version, it has added a new parameter for the size of the comments batch that were going to be analysed, because in the original version this size was fixed and was defined in a configuration file. This way, whatever the size of the comments batch, the machine learning model will work correctly and perform the desired analysis (Code 6).

```python
def create_dataloader(params, tweets, teams) -> dict:
    # code here…

    data_loader = DataLoader(
        dataset=dataset,
        batch_size=params["model"]["batch_size"],
        shuffle=False,
    )
    return data_loader
```

Original Code

```python
def create_dataloader(params, tweets, teams, batch_size) -> dict:
    # code here…


    data_loader = DataLoader(
        dataset=dataset,
        batch_size=batch_size,
        shuffle=False,
    )
    return data_loader
```

Modified Code

Code 6 - Comparison of the original code and the modified code in the create_dataloader method of the data.py file

In the inference.py file, that is, in the file called by the Kafka structure and the file that executes the NLP sentiment analysis model. A method called inference was created to be called by Kafka, within this is all the execution of the file. Due to the limitations and characteristics of the server, it was necessary to modify how inference occurs so that it is executed only by CPU as can be seen in Code 7. Also in this method, that the method mentioned above, create_dataloader, is called and receives the new batch size parameter.

```
checkpoint =
torch.load(params["model"]["model_checkpo
int"])
```

Original Code

```
checkpoint = torch.load(
        params["model"]["model_checkpoi
nt"], map_location=torch.device('cpu'))
```

Modified Code

Code 7 - Changes made to run the model only by CPU

This is where the biggest difference from the original occurs. After the inference values are received, they are placed in an array and normalized through a SoftMax function, then an average of each of the three sentiments (negative, neutral, and positive) is calculated. Finally, these are labelled with the corresponding sentiment and returned to Kafka (Code 8). This change occurs so that the returned result is from a batch of comments and not from each comment. There is also a change between the first approaches and the final approach (Code 8) of the Kafka structure, where the returned values went from being the number of comments that had each sentiment as predominant, to the real value that each comment had of each sentiment and in the end an average of these values. This way, comments that had sentiments with similar values, but only the one that was slightly larger than the others was returned, now have all values counted in the final result.

```
predictions = [labels[score.argmax()]
for score in pred_scores[0]]

print(predictions)
```

Original Code

```
    logits_array =
np.array(pred_scores[0])
    probabilities_array =
softmax(logits_array)
    average_probabilities =
np.mean(probabilities_array, axis=0)

    x = np.array(average_probabilities,
dtype=np.float32)
    x_list = x.tolist()

    result = dict(zip(labels, x_list))
    return result
```

Modified Code (final approach)

Code 8 - Modifications to the values that Kafka Structure will receive

5.2 Topic Analysis Model

In the inference.py file, a method with the same name was created to be called by Kafka. Within this method is all the execution of the inference and the changes made to this model. The first change was to alter the return data from the topic of each comment to the percentage of the number of comments in each topic. As described in previous changes, this allows all values from all topics in all comments to be used to calculate the final result (Code 11), instead of just print and show the results of each comment.

This approach had a problem, because did not correspond to the feeling and themes of the comments, or at least there were some comments where a discrepancy was noticeable and that deteriorated the finals results.

As result of the problem described in the last paragraph, the last approach suffers another change, to send the average of the value attributed to each topic (Code 9).

```
output = wstc.predict(input, 0)

prediction = [label2name[pred] for pred
in output]

print(prediction)
```

Original Code

```
output = wstc.predict(input, 0)

result = {label2name[i]:   sum(array[i]
for array in output)/len(output) for i
in range(len(label2name))}

return result
```

Modified Code (final approach)

Code 9 - Comparison between the original code and the final modified code of the topic analysis model

Like the changes produced in 5.1 Sentiment Analysis Model for the final approach, here the changes were the same: Change the percentage of comments with each topic as predominant, to an average of the value that each comment has for each topic. The topic analysis model assigns a value to each topic in each comment. The change made corresponds to taking an average of each value of each topic and returning it, as seen in Code 9.

5.3 Results

In this sub-chapter the results of the differences made to the original NLP models, will be showed.

In Figure 34, is possible to see the comments used to show the differences in the results of the changes made to the sentiment analysis model and topic analysis model.

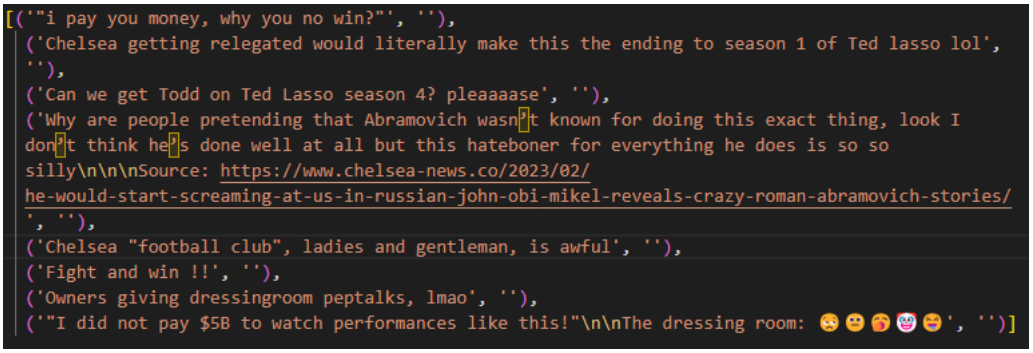

Figure 34 - Data used to test first change in the sentiment analysis model

5.3.1 Sentiment Analysis

The original code only returned the sentiment of each comment it was sent to analyze, but for the project it was necessary to get results for a batch of comments each time the model was called to execute. So, a change was made to return the percentage of comments that had each sentiment.

Using the comments of the Figure 34 and the developed code described above, the results were the ones in Table 3.

Table 3 - Results of first change in sentiment analysis model

Sentiment	Count	Percentage
NEUTRAL	6	75.0
POSITIVE	1	12.5
NEGATIVE	1	12.5

The values returned are not the best ones for this type of project, because if a comment I very close to positive, but the value attributed to the neutral sentiment is slightly higher, the model will return that the comment is neutral. Doing the changes described in 5.1 allows that every value of every sentiment of each comment is considered. The results can be seen in Table 4.

Table 4 - Results of second change in sentiment analysis model

Sentiment	Real Value Sentiment
NEUTRAL	0.7441776990890503
POSITIVE	0.1411905586719513
NEGATIVE	0.11463174968957901

5.3.2 Topic Analysis

The original code only returned the main topic of each comment, but for the project it was necessary to get results for a batch of comments. So, a change was made to return the percentage of comments that had each topic. Using the comments of the Figure 34**Error! Reference source not found.** and the developed code described above, the results were the ones in Table 5.

Table 5 - Results of first change in topic analysis model

Topic	Percentage
'kickoff'	87.5
'red_card'	12.5

The values returned are not ideal for this type of project. Each comment has multiple topics, but the way the models were done, if a topic is the more predominant by a slightly margin it will be the only one returned for that comment. Doing the changes described in Topic Analysis Model5.2 allows that every value of every topic of each comment is considered. The results can be seen in Table 6.

Table 6 - Results of final change in topic analysis model

Topic	Real Value
"save"	0.04784945445135236
"foul"	0.007321778335608542
"free_kick"	0.01275465206708759
"goal"	0.041509130503982306
"kickoff"	0.308865187689662
"offside"	0.0333012321498245
"penalty"	0.033018454210832715
"red_card"	0.2416600715368986
"yellow_card"	0.04973530350252986
"back_pass"	0.19243361987173557
"substitution"	0.03155109751969576

6 Results of the Pilot Test

In this chapter, the performance results of the pilot test, carried out by MOG Technologies, will be presented. The results of the NLP models, that is, the sentiment analysis and topic analysis results will not be discussed because that is not inside this document theme. This test was carried out in a real environment, that is, with the platform that the customers would use and with a real event in real-time. On the ISEP side, all the settings would also be real. Thus, this was the final test of validation of the project and that proved all the functioning and viability of the developed content. The chosen event was a sports event, namely the final of the Conference League[24], a men's football competition, held by UEFA[25], the main organization of European football. The event took place on Wednesday, June 7, 2023, at 8 pm (Portugal mainland time). The match took place at the Fortuna Arena stadium in Prague and faced the Fiorentina[26] team and the West Ham[27] team. The West Ham team won 2 – 1 (*Fiorentina 1-2 West Ham: Bowen Arrows Hammers to Europa Conference League Final Glory | UEFA Europa Conference League | UEFA.Com*, n.d.).

The match had an attendance of 17363 (*ACF Fiorentina - West Ham United 1:2 (Europa Conference League 2022/2023, Final)*, n.d.) and many millions watched on tv. Unfortunately, there is no official number provided by UEFA, but being a recent competition, created for the 2021-2022 season as the third competition in the hierarchy of European competitions (*UEFA Executive Committee Approves New Club Competition | Inside UEFA | UEFA.Com*, n.d.), it is possible to assume that its audience was much smaller than the audience of the men's

[24] https://www.uefa.com/uefaeuropaconferenceleague/
[25] https://www.uefa.com/
[26] https://www.acffiorentina.com/en
[27] https://www.whufc.com/

Champions League[28] final. This had, at its maximum period, 450 million people watching simultaneously (*TV Ratings: How Many People Watched the 2023 Champions League Final? - AS USA*, n.d.). Due to having a very large audience, but at the same time not being huge, as in other competitions, it was the perfect event to test the capabilities of all parts of the product developed in a real environment.

6.1 Description of the results

This test was done using the latest approach of the Kafka architecture, as described in 4.6 and using multi-processing with simultaneously getting the data from the social networks. This was used because was the one approach with better results, as demonstrated in 4.8.4.

6.1.1 YouTube

Here, the process described in 4.7.2 was used. As the Chat Downloader requires sending a URL to get comments from it, the chosen livestream has the URL: "https://www.youtube.com/watch?v=tJw3pkKZ-Wc" and has more than 109500 views. This livestream is from the channel "L' immigré parisien"[29], a channel in French language.

In total 220 analysis were carried out, 110 of each, where 7910 comments were used to those analysis. The maximum number of comments analysed in one iteration was 76 and the minimum was 63.

In Figure 35 is possible to see the number of comments (left y axis), represented by the blue bars, in each iteration (x axis) and the time that has pass from the previous iteration (right y axis), represented by the red line. From this graphic, that represents the sentiment analysis values, it is possible to verify that the number of comments was constant, having varied at most by 13 comments. It is also possible to see that in the first 50 iterations there is no obvious pattern in the time that passes between each iteration even though the comments are in similar quantities. From the fiftieth iteration, a pattern begins to emerge, where the time is being constant during 5 or 6 iterations, and then the time interval increases and right after that decreases, but this value is not dependent on the number of comments analysed. Factors that contribute to this lack of connection may be the total amount of analyses that are being executed at those moments or the time that was necessary to obtain the comments from YouTube and their size.

[28] https://www.uefa.com/uefachampionsleague/
[29] https://www.youtube.com/@LImmigreparisien

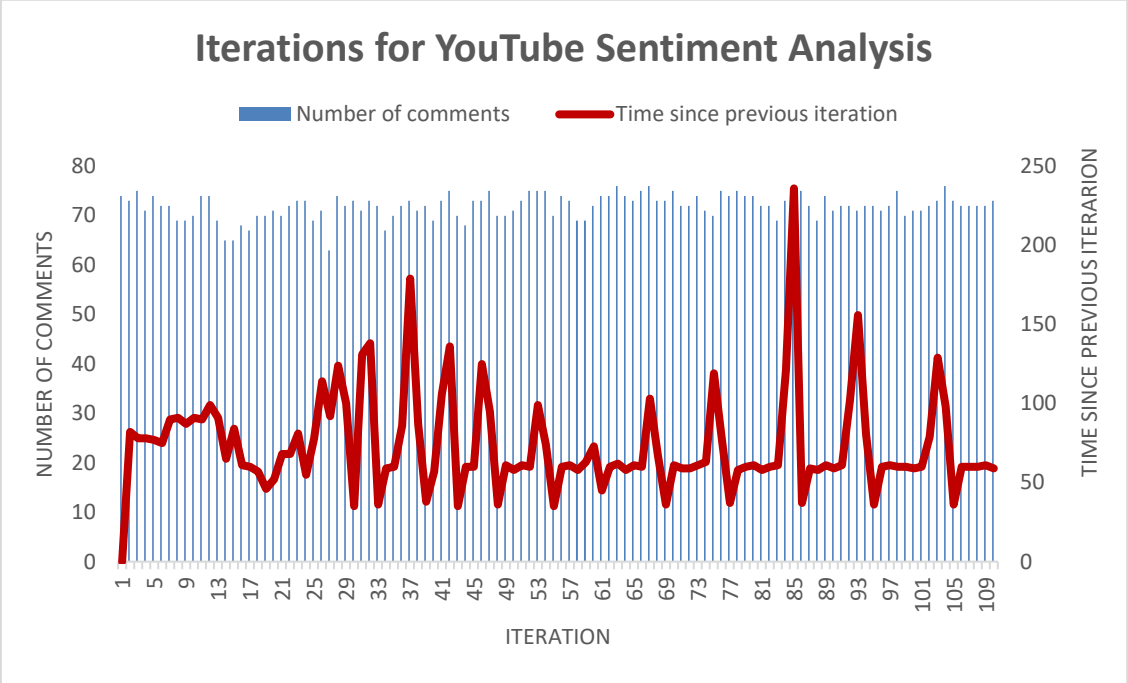

Figure 35 - Iterations for YouTube Sentiment Analysis

In Figure 36 is possible to see the number of comments (left y axis), represented by the blue bars, in each iteration (x axis) and the time that has pass from the previous iteration (right y axis), represented by the red line. This graphic represents the topic analysis. It is possible to observe that the pattern of the graph is very similar to Figure 35, that is, where an analysis takes longer, another one happens the same. And this delay is not related to the number of YouTube comments.

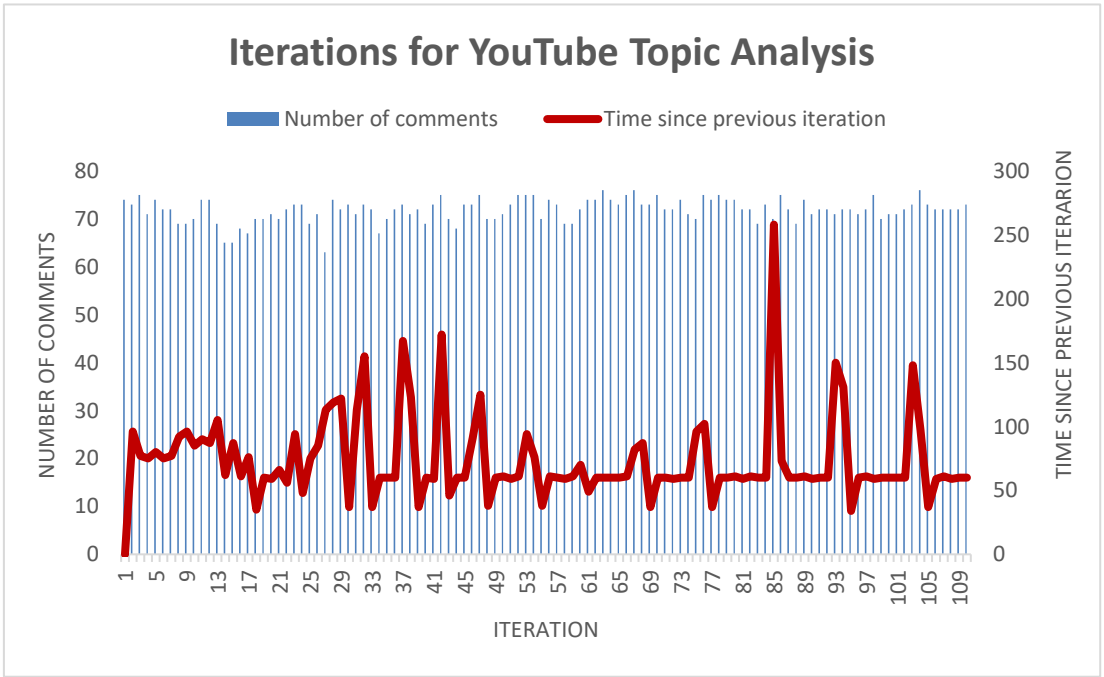

Figure 36 - Iterations for YouTube Topic Analysis

By comparing Figure 35 and Figure 35 , it can be seen that the patterns of the total execution times of the analyses are similar. Despite this similarity, sentiment analysis was on average faster than topic analysis. The first took an average of 73.37 seconds, while the second took 73.55 seconds on average. It is a residual difference. It should be noted that these temporal values do not correspond only to the inference performed by the NLP models, but rather to all the time that passes between the beginning of an iteration and the beginning of the next, that is, between the moment when the data start to be obtained until sending the response of the analysis results via Kafka to MOG.

6.1.2 Reddit

As described in 4.7.1, PRAW was used to obtain reddit comments. The URL of the event analysed was "https://www.reddit.com/r/soccer/comments/143l52x/match_thread_fiorentina_vs_west_ham_united_europa/". This reddit post currently has more than 5200 comments, but a total of 5475 comments were analysed during the test. The current number may be lower if, in the meantime, comments were deleted or removed from the post. A total of 94 analyses were performed, 47 sentiment analyses and 47 topic analyses. The maximum number of comments analysed in one iteration was 300. This value was received through the request Kafka message, if it wasn't mentioned, the max amount would be 200.

In Figure 37 is visible the number of comments (left y axis), represented by the blue bars, each iteration (x axis) and the time that has pass since the previous iteration (right y axis), represented by the red line. This graphic illustrates the sentiment analysis values, it is possible to verify that the number of comments wasn't constant unlike in the YouTube. Here there were iterations between 8 and 300 comments. It is also possible to see that in the first 25 iterations the number of comments was small, this time correspond to the beginning of the match, but from 26 iteration until the end the number of comments rise drastically. In the reddit analysis, and unlike the YouTube, almost every time the number of comments raised, the time spent on the analysis also raised.

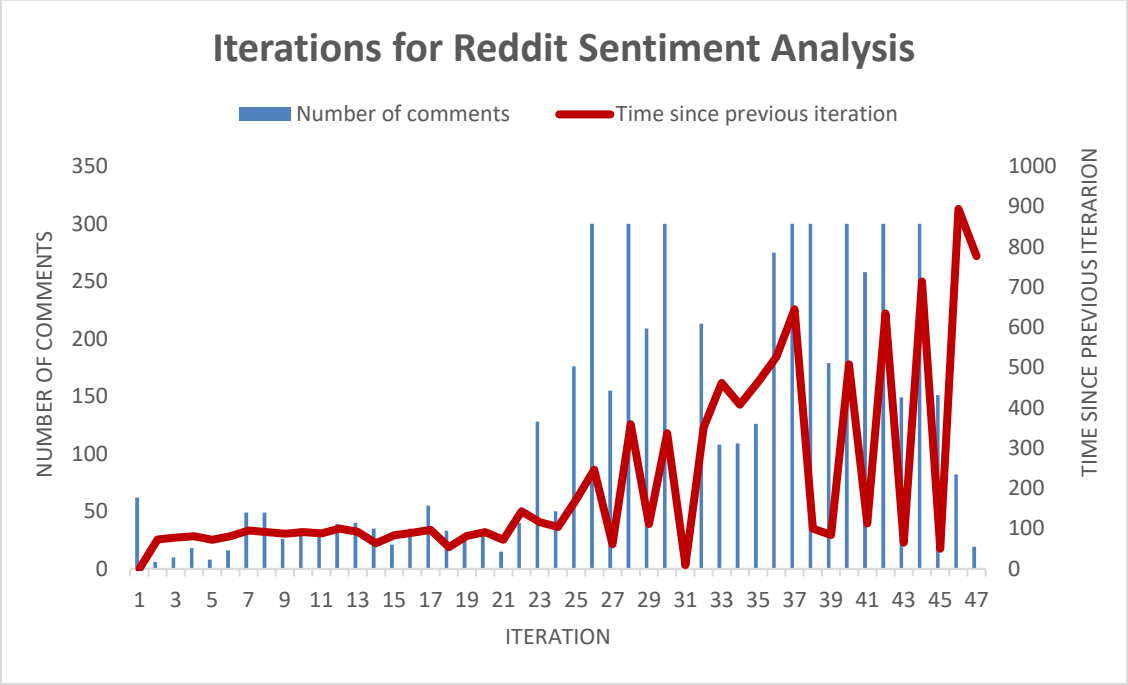

Figure 37 - Iterations for Reddit Sentiment Analysis

The Figure 38 is a graphic that in the left y axis has the number of comments, represented by the blue bars, on the x axis has the number of iteration, and the time that has pass from the previous iteration on the right y axis, represented by the red line. This graphic represents the topic analysis. It is possible to observe that the pattern of the graph is very similar to Figure 37, that is, where an analysis takes longer, another one happens the same. Equal to the sentiment analysis, almost every time the number of comments raised, the time spent on the analysis also raised.

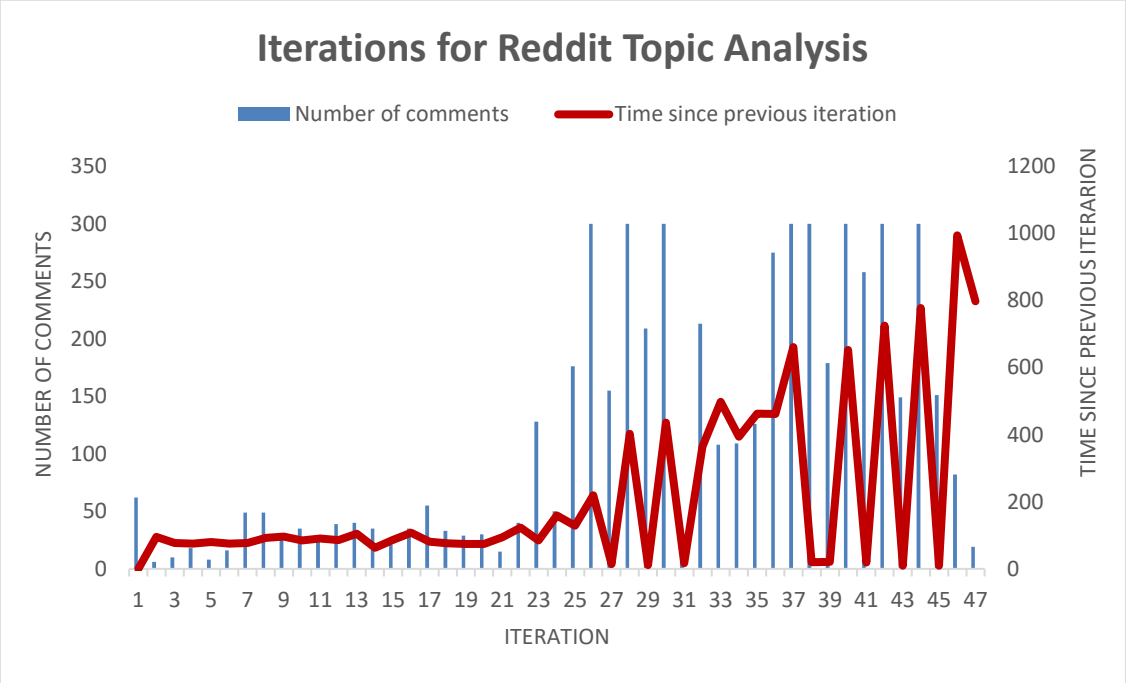

Figure 38 - Iterations for Reddit Topic Analysis

When comparing Figure 37 and Figure 38, it can be seen that the patterns of the total execution times of the analyses are similar. Despite this similarity, sentiment analysis was on average faster than topic analysis. The first took an average of 215.19 seconds, while the second took 215.96 seconds on average. It is a residual difference. It should be noted that these temporal values do not correspond only to the inference performed by the NLP models, but rather to all the time that passes between the beginning of an iteration and the beginning of the next, that is, between the moment when the data start to be obtained until sending the response of the analysis results via Kafka to MOG.

Contrary to what happened with YouTube, in the reddit, the amount of comments to be analysed had a direct influence on the performance of the models, but not only that but also the presumption that the size of each comments in reddit are bigger that the ones on YouTube.

6.1.3 Internal Chat

In the case of the chat comments, they were sent in each Kafka message, in order to be analysed. Throughout the entire test execution, 10 messages were received to analyse chat comments, in total 15 comments were analysed. This means that there was little movement on the chat platform. The highest number of comments in a request was only 3 and the lowest was 1 (Figure 39).

In Figure 40 and Figure 39, are expressed some values about the sentiment analysis and topic analysis of the chat comments, respectively. In the left y axis, represented by the silver bars, are the number of comments in each message. In the right y axis, represented by the black line,

is the number of second that have pass since the last analysis (previous iteration). The x axis represents the iteration, that is, the identification of analysis, in this case like only were 10 messages to analyse comments, the maximum value is 10. Is possible to see that the most time that pass between analysis were between the seventh and eighth analysis, but unlike the Reddit and YouTube results, this one's aren't very meaningful because each analysis is not depending on the other or in obtaining comments while running, that's why the values represented in the graphics are the equal. Here the main values, are the amount of time each analysis took, especially because there were only a few comments.

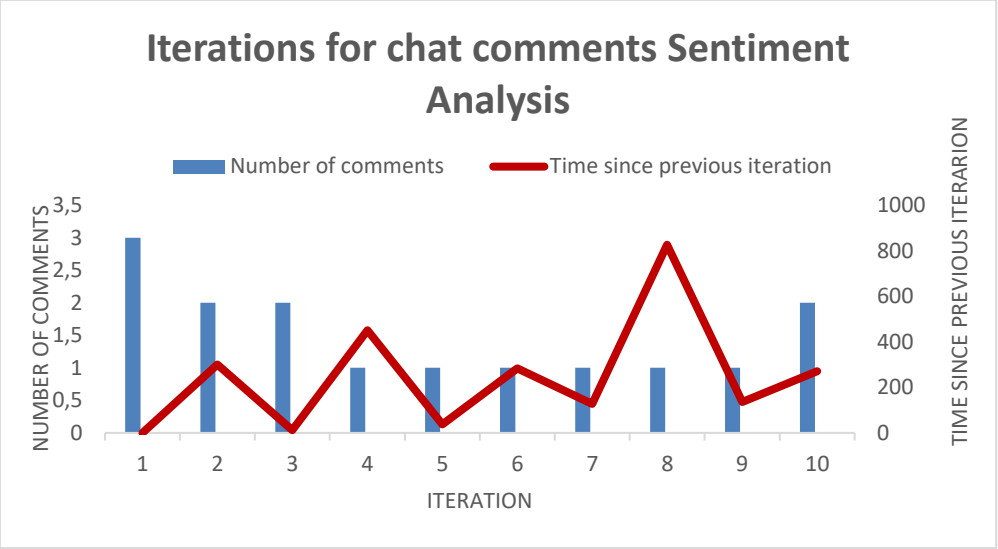

Figure 40 - Iterations for chat comments Sentiment Analysis

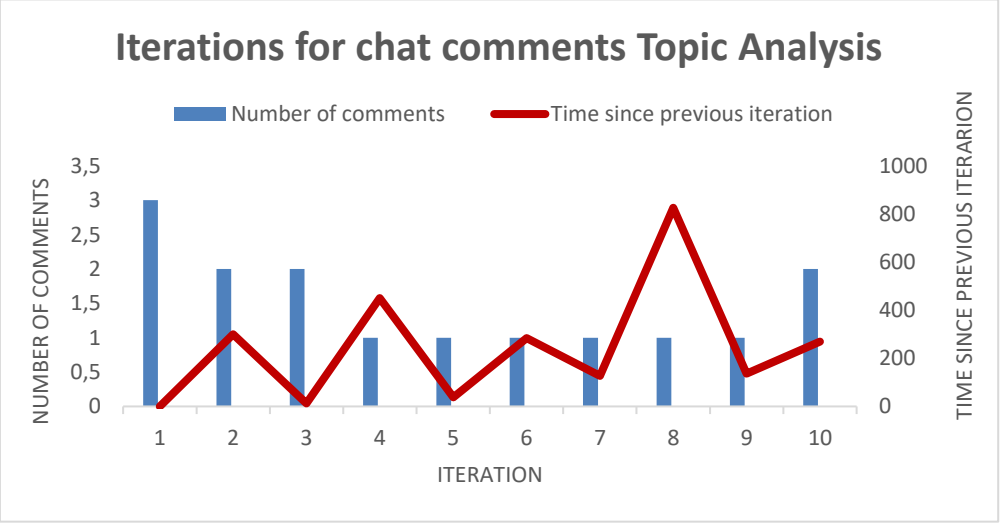

Figure 39 - Iterations for chat comments Topic Analysis

From the timestamps of this test was possible to create the where the seconds that each analysis took to be executed are represented. When analysing the data from the graphic, it is possible to verify that the nineth iteration was the one that had the fastest to analyse, 5 seconds each, while the eighth iteration had the slowest analyses, with 47 seconds each. In both iterations, the number of comments analysed was only one, so it is not possible to make a direct relationship between the number of comments and the analysis time. Here other factors enter, such as the size of the comments or the amount of analyses that were being executed in parallel. The more analyses, the more hardware resources will be required.

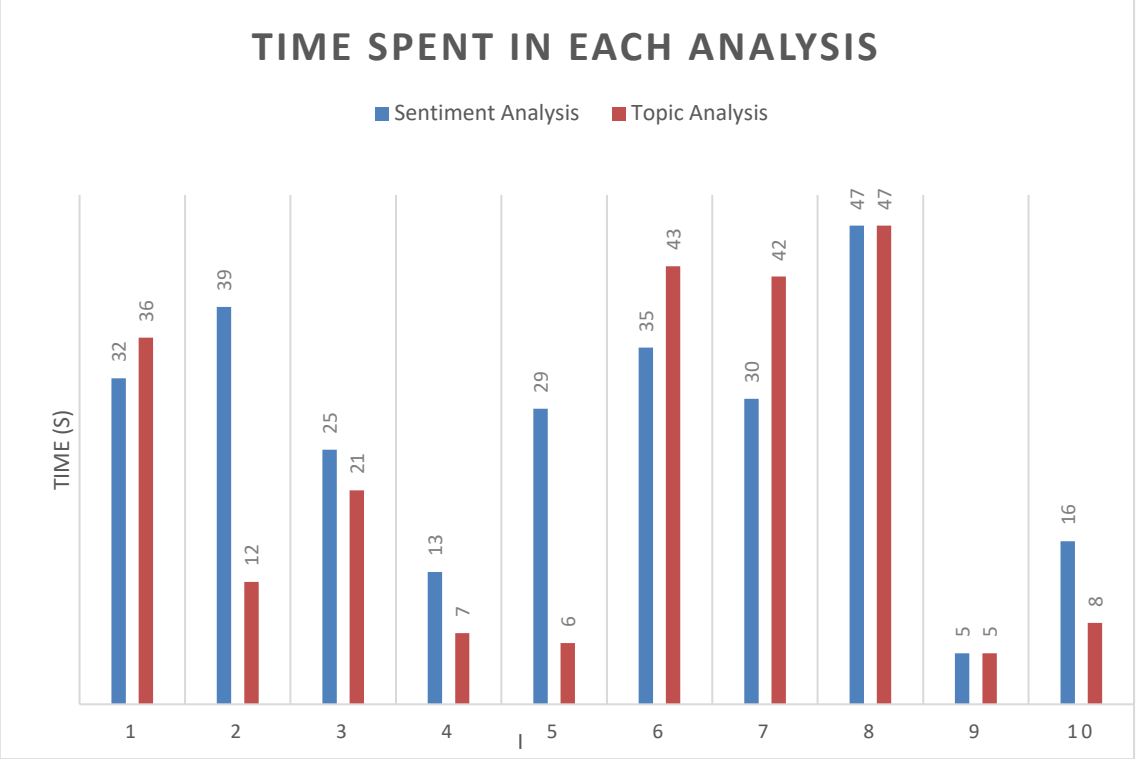

Figure 41 - Time spent in each analysis for chat comments

It is also possible to verify that the sentiment analysis was the most time-consuming in five occasions, while the topic analysis was only the most time-consuming in three, having in the other two iterations the same execution time. On average, the sentiment analysis took 27.1 seconds, and the topic analysis took 22.7 seconds, that is, 4.4 seconds faster.

7 Conclusion

On this chapter, a summary of the developed work is presented and its conclusions. Additionally, the improvements that can be made are also described, that come from a set of limitations. Future work to be done is also laid out.

7.1 Summary and conclusions

The project had as its main objective the creation of a multimodal reactive system for media transmission, whose target audience was not only media and audiovisual companies, but also the clubs themselves. The system had to have several important features, such as being a cloud system customized for each user, an integrated system capable of generating real-time analytical data from social network information. And finally, it was also necessary that it created summarization content from the analyses and information obtained.

The mains goals of this thesis were:

- Create a hosting structure for de NLP models, capable of execute them without issues.
- Ensure the communication between MOG servers and ISEP servers and to that a creation of a communication system.
- Developed a statistical model of summarization of the results originating from the machine learning models.

The first two goals were mandatory and were conclude with success as can be seen previously in this document, but the third goal was not developed and is one of the future works proposed. That said the PLAYOFF project did not miss the statistical summarization, because the front-end tool from MOG has those capabilities.

To choose which communication framework to use to develop the project, research and a comparison between Apache Kafka and RabbitMQ was done. Apache Kafka was the chosen one

because applications registered to a topic and are notified in real time every time new messages arrive to the broker, which allows quickly processing of messages. The infrastructure with Kafka can be easily scaled horizontally and the messages can persist for a period of time instead of been deleted as soon as they reach the destination application. Has a huge user base and allows the implementation of secure protocols in the communications. This security protocols were a option, but in the end weren't used and are recommended as future work.

As a Kafka structure exchanges messages and needs topics, it was necessary to define these for the system to be developed. The following topics of the request messages that MOG would send to ISEP were defined: "master_topic" for the request messages. In the case of the messages from ISEP to MOG, the topics were: "twitter_SA", "twitter_TA" "reddit_SA", "reddit_TA", "chat_SA", "chat_TA", for the response messages with results where, each topic refers to the social network and the type of analysis performed, and "feedback" for the messages to confirm the receipt of a new request. Later was needed to change the "twitter_SA" and "twitter_TA" topics to "youtube_SA" and "youtube_TA" when the change of using YouTube instead of Twitter was made.

There also different formats of the messages being exchange in Kafka. In the final version, it was determined that the MOG to SEP messages would have the following parameters: "event_id", which indicates the id of the event, "timestamp", instant of time, in Unix, of the request, "channel", which corresponds to the URL or hashtag that will undergo the analysis, "limit", maximum number of comments to analyse in each iteration, "analysis_status", whether it is to start a processing or to end one that is in progress, "kafka_topics", where the types of analysis and social network to process are indicated, "chat_messages", in case it is to analyse messages from the chat platform. The messages from ISEP to MOG have the "event_id", the "timestamp", "channel", "emotions" (where the results of the sentiment analysis are) or "topics" (where the results of the topic analysis are) and "n", that indicates how much comments were analysed.

The system architecture went through 4 iterations, similar to each other. In the final approach, a MOG producer sends a message requesting the start of analysis. This message is received in the "master_consumer" of ISEP, which coordinates the entire execution. It receives the message and checks if it is a message to end an execution already in progress or if it is to start a new one. Started a new processing, the comments of the intended event of the indicated social network are obtained and then the inference with these data is started. When the models return the results, a message is sent by the ISEP producer to the consumer in MOG with these results.

To run multiple analysis ate the same time was necessary to implement a concurrence solution, which means it was necessary to choose between threads and multi-processes. The first implementation was with threads. The results were good with the sentiment analysis always take more time than the topic analysis. With YouTube comments it took, on average, 26.6 seconds against 6.5 seconds. With Reddit comments it took 71.2 seconds against 8.2 seconds, and with chat comments it was 12.31 times faster. In this approach only when both analyses were finished, a new search for new comments was started. The next approach was with multi-

processes and like the one with threads, only when both analyses were finished, a new search for comments started (sequential approach). With the YouTube comments, it took the sentiment analysis model 73.4 seconds on average and the topic analysis model 18.2 to finished. With Reddit comments it took 73.5 seconds and 20 seconds, on average, to the sentiment analysis and topic analysis models, respectively. Lastly, with the chat it took around 5.37 times more to run the sentiment analysis model. The final approach was with multi-threads, but instead of the data being search after the analyses ended, it was always getting new data concurrently with the analysis (parallel approach). This, in theory, would help the total time of execution and ate the same time allows for more analysis. Sentiment analysis took 68 seconds, against 12.7 seconds of topic analysis with YouTube comments. With Reddit, it took 71.9 seconds, against 11.6 seconds and with chat the topic analysis was 4.5 times faster than the sentiment analysis. But the main take from those tests, are that the parallel approach was the best one, like the comparison in 4.8.5 allows to see. The threads approach took 2054 seconds in total, the sequential multi-processes approach took 2720 seconds, and the parallel multi-processes took 2011. The difference between the last one and the others would be even greater if there were more analysis ate the same time.

Besides the communication structure, was also needed to do a few changes on the NLP models, so that they could run on the environment available and that the results were according to the intended. Both models were made to return the results of comments individually, but the project require to send a batch of comments and the results be aggregated to that batch. With that in mind, the first change was just to return an average of the results of each comment, but analysing the results values was perceptible that some results were not the most reliable. This happened because both models only return the sentiment or topic of the predominant one, so if two sentiment have similar values, but one is slightly higher it would only return that, and with the topics analysis model the same occurred, and the other values were “lost”. To resolve this a new change was made, instead of returning the average of the results, it would return the average of each sentiment in each comment and the average of each topic in each comment. This way every value calculated by the models were considered and returned to MOG.

In conclusion, the mandatory objectives were successfully completed, and the concept was proven possible as the Pilot Test, done in real time environment demonstrate. This dissertation resulted in one scientific publication (Multimodal sports media transmission environment - Playoff Project), that have been presented at a scientific conference (WCASET-2023: World Conference on Applied Science, Engineering and Technology, on July 28, 2023 in Athens).

7.2 Future Work

As future work on NLP models, this work has already been described in (Ferraz Barbosa Soares de Albergaria, 2022), but there are improvements to be made, specifically in the topic analysis model to better determine topics in shorter text forms.

The main goal of future work is to make the summarization statistical model with the results provide by the structure developed and described in this thesis. To do this, more data from the social media APIs can be retrieved, like gender, age, location, etc. This way is possible to create a robust modal to complement everything developed until this point.

For the communication tool, the main topic to be developed is the introduction of Kafka security mechanisms, namely the “SASL+/SCRAM-SHA-512” protocol and verifying how the introduction of security in Kafka communication affects exchange performance of information. In addition to the security mechanisms, other improvements can also be tested:

- o Use of more than one producer to manage sent messages.
- o Test more social networks within the system and check the results and performance.

Another improvement that is worth exploring in the future is the merging of NLP models with Kafka. Usually, the training and deployment process are two separate processes, but they can be together and present as advantages (Waehner, 2019):

1. Better performance in terms of runtime.
2. Such will be more reduced.
3. Features prebuilt in the development tool will facilitate the development of the models.

References

8 Applications of Sentiment Analysis. (n.d.). Retrieved July 31, 2023, from https://monkeylearn.com/blog/sentiment-analysis-applications/

ACF Fiorentina - West Ham United 1:2 (Europa Conference League 2022/2023, Final). (n.d.). Retrieved September 25, 2023, from https://www.worldfootball.net/report/europa-conference-league-2022-2023-finale-acf-fiorentina-west-ham-united/

Ajith, P. (2020, August 8). *Spark Twitter Streaming: A Real Time Twitter data streaming ETL Pipeline and Sentiment Analyser*. https://github.com/pran4ajith/spark-twitter-streaming

Alghamdi, R., & Alfalqi, K. (2015). A Survey of Topic Modeling in Text Mining. *International Journal of Advanced Computer Science and Applications*, *6*(1). https://doi.org/10.14569/IJACSA.2015.060121

Apache Kafka. (n.d.). Retrieved July 28, 2023, from https://kafka.apache.org/documentation/

Apache Kafka - Security Overview. (n.d.). Retrieved June 28, 2023, from https://kafka.apache.org/documentation/#security_overview

Apache ZooKeeper. (n.d.). Retrieved July 28, 2023, from https://zookeeper.apache.org/

Applications of Sentiment Analysis - DataScienceCentral.com. (n.d.). Retrieved July 31, 2023, from https://www.datasciencecentral.com/application-of-sentiment-analysis/

Atanassova, I., Bertin, M., & Mayr, P. (2019). Mining Scientific Papers: NLP-enhanced Bibliometrics. *Frontiers in Research Metrics and Analytics*, *4*. https://doi.org/10.3389/FRMA.2019.00002

Belle Wong, J. D. (2023, May 18). *Top Social Media Statistics And Trends Of 2023 – Forbes Advisor*. https://www.forbes.com/advisor/business/social-media-statistics/

Benjamins, R. (2020). A choices framework for the responsible use of AI. *AI and Ethics 2020 1:1*, *1*(1), 49–53. https://doi.org/10.1007/S43681-020-00012-5

Bik, H. M., & Goldstein, M. C. (2013). An Introduction to Social Media for Scientists. *PLOS Biology*, *11*(4), e1001535. https://doi.org/10.1371/JOURNAL.PBIO.1001535

Billings, A. C., Broussard, R. M., Xu, Q., & Xu, M. (2018). Untangling International Sport Social Media Use: Contrasting U.S. and Chinese Uses and Gratifications Across Four Platforms. *Https://Doi.Org/10.1177/2167479518790014*, *7*(5), 630–652. https://doi.org/10.1177/2167479518790014

Chawinga, W. D. (2017). Taking social media to a university classroom: teaching and learning using Twitter and blogs. *International Journal of Educational Technology in Higher Education*, *14*(1), 1–19. https://doi.org/10.1186/S41239-017-0041-6/TABLES/5

Chen, P.-Y. (2021, December 15). *What is AI adversarial robustness? | IBM Research Blog*. https://research.ibm.com/blog/securing-ai-workflows-with-adversarial-robustness

Cihon, P., Schuett, J., & Baum, S. D. (2021). Corporate Governance of Artificial Intelligence in the Public Interest. *Information 2021, Vol. 12, Page 275*, *12*(7), 275. https://doi.org/10.3390/INFO12070275

Cohan, A., & Goharian, N. (2018). Scientific document summarization via citation contextualization and scientific discourse. *International Journal on Digital Libraries*, *19*(2–3), 287–303. https://doi.org/10.1007/S00799-017-0216-8/METRICS

Devika, M. D., Sunitha, C., & Ganesh, A. (2016). Sentiment Analysis: A Comparative Study on Different Approaches. *Procedia Computer Science*, *87*, 44–49. https://doi.org/10.1016/J.PROCS.2016.05.124

Dhar, A., Mukherjee, H., Dash, N. S., & Roy, K. (2021). Text categorization: past and present. *Artificial Intelligence Review*, *54*(4), 3007–3054. https://doi.org/10.1007/S10462-020-09919-1/METRICS

Dijkstra, E. W., Lamport, ; L, Owicki, ; S, & Gries, D. (1979). Verifying properties of parallel programs: an axiomatic approach. *IEEE TRANSACTIONS ON COMPUTERS*, *1*(9), 536–552.

Dixon, S. (2023, July 11). *Social Media Statistics & Facts | Statista*. https://www.statista.com/topics/1164/social-networks/#topicOverview

Dizikes, P. (2020, September 24). *Why social media has changed the world — and how to fix it | MIT News | Massachusetts Institute of Technology*. MIT News. https://news.mit.edu/2020/hype-machine-book-aral-0924

Dobbelaere, P., & Esmaili, K. S. (2017). Industry paper: Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations. *DEBS 2017 - Proceedings of the 11th ACM International Conference on Distributed Event-Based Systems*, 227–238. https://doi.org/10.1145/3093742.3093908

Drahošová, M., & Balco, P. (2017). The analysis of advantages and disadvantages of use of social media in European Union. *Procedia Computer Science*, *109*, 1005–1009. https://doi.org/10.1016/J.PROCS.2017.05.446

Drus, Z., & Khalid, H. (2019). Sentiment Analysis in Social Media and Its Application: Systematic Literature Review. *Procedia Computer Science*, *161*, 707–714. https://doi.org/10.1016/J.PROCS.2019.11.174

Fergus, P., & Chalmers, C. (2022). *Deploying and Hosting Machine Learning Models*. 299–317. https://doi.org/10.1007/978-3-031-04420-5_13

Ferraz Barbosa Soares de Albergaria, M. (2022). *Near Real-Time Sentiment and Topic Analysis of Sport Events*.

Filo, K., Lock, D., & Karg, A. (2014). Sport and social media research: A review. *Https://Doi.Org/10.1016/j.Smr.2014.11.001*, *18*(2), 166–181. https://doi.org/10.1016/J.SMR.2014.11.001

Fiorentina 1-2 West Ham: Bowen arrows Hammers to Europa Conference League final glory | UEFA Europa Conference League | UEFA.com. (n.d.). Retrieved September 25, 2023, from https://www.uefa.com/uefaeuropaconferenceleague/news/0282-1833ad184f9c-3bd42a7f4d8e-1000/

Goodhope, K., Koshy, J., Kreps, J., Narkhede, N., Park, R., Rao, J., & Ye, V. Y. (2012). Building LinkedIn's Real-time Activity Data Pipeline. *IEEE Data Eng. Bull.*, *35*, 33–45.

Heymann, H., Kies, A. D., Frye, M., Schmitt, R. H., & Boza, A. (2022). Guideline for Deployment of Machine Learning Models for Predictive Quality in Production. *Procedia CIRP*, *107*, 815–820. https://doi.org/10.1016/J.PROCIR.2022.05.068

Heymann, H., Mende, H., Frye, M., & Schmitt, R. H. (2023). Assessment Framework for Deployability of Machine Learning Models in Production. *Procedia CIRP*, *118*, 32–37. https://doi.org/10.1016/J.PROCIR.2023.06.007

Hong, Z., Ward, L., Chard, K., Blaiszik, B., & Foster, I. (2021). Challenges and Advances in Information Extraction from Scientific Literature: a Review. *JOM*, *73*(11), 3383–3400. https://doi.org/10.1007/S11837-021-04902-9/METRICS

Hu, Y., Kuang, W., Qin, Z., Li, K., Zhang, J., Gao, Y., Li, W., & Li, K. (2021). Artificial Intelligence Security: Threats and Countermeasures. *ACM Computing Surveys (CSUR)*, *55*(1). https://doi.org/10.1145/3487890

Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, *8*(1), 216–225. https://doi.org/10.1609/ICWSM.V8I1.14550

Ibáñez, J. C., & Olmeda, M. V. (2022). Operationalising AI ethics: how are companies bridging the gap between practice and principles? An exploratory study. *AI and Society*, *37*(4), 1663–1687. https://doi.org/10.1007/S00146-021-01267-0/TABLES/9

Indhraom Prabha, M., & Umarani Srikanth, G. (2019). Survey of Sentiment Analysis Using Deep Learning Techniques. *Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICIICT 2019*. https://doi.org/10.1109/ICIICT1.2019.8741438

Isayev, O. (2019). Text mining facilitates materials discovery. *Nature 2021 571:7763*, *571*(7763), 42–43. https://doi.org/10.1038/d41586-019-01978-x

Jones, M. (2015, June 16). *History of Social Media: The Invention of Online Networking*. https://historycooperative.org/the-history-of-social-media/

Kafka vs RabbitMQ - A Head-to-Head Comparison for 2023. (n.d.). Retrieved July 28, 2023, from https://www.projectpro.io/article/kafka-vs-rabbitmq/451#mcetoc_1fb64j0f71

Kherwa, P., & Bansal, P. (2019). Topic Modeling: A Comprehensive Review. *EAI Endorsed Transactions on Scalable Information Systems*, "*7*"(24), 1–16. https://doi.org/10.4108/EAI.13-7-2018.159623

Knoke, D. (2014). Origins of Social Network Analysis. *Encyclopedia of Social Network Analysis and Mining*, 1229–1233. https://doi.org/10.1007/978-1-4614-6170-8_362

Launch of RabbitMQ Open Source Enterprise Messaging. (n.d.). Retrieved July 28, 2023, from https://www.rabbitmq.com/resources/RabbitMQ_PressRelease_080207.pdf

Lee, E. A. (2006). *The Problem with Threads*. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.html

Li, B., Dittmore, S. W., Scott, O. K. M., Lo, W. juo, & Stokowski, S. (2019). Why we follow: Examining motivational differences in following sport organizations on Twitter and Weibo. *Sport Management Review*, *22*(3), 335–347. https://doi.org/10.1016/J.SMR.2018.04.006

Lin, Y. S., Jiang, J. Y., & Lee, S. J. (2014). A similarity measure for text classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *26*(7), 1575–1590. https://doi.org/10.1109/TKDE.2013.19

López-Carril, S., Escamilla-Fajardo, P., González-Serrano, M. H., Ratten, V., & González-García, R. J. (2020). The Rise of Social Media in Sport: A Bibliometric Analysis. *Https://Doi.Org/10.1142/S0219877020500418*, *17*(6). https://doi.org/10.1142/S0219877020500418

Lorè, F., Lorè, L., Basile, P., Appice, A., Marco De Gemmis, ·, Donato Malerba, ·, & Semeraro, G. (2023). An AI framework to support decisions on GDPR compliance. *Journal of Intelligent Information Systems 2023*, 1–28. https://doi.org/10.1007/S10844-023-00782-4

Lu, Y. (2019). Artificial intelligence: a survey on evolution, models, applications and future trends. *Https://Doi.Org/10.1080/23270012.2019.1570365*, *6*(1), 1–29. https://doi.org/10.1080/23270012.2019.1570365

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16–32. https://doi.org/10.1016/J.COSREV.2017.10.002

Mäntymäki, M., Minkkinen, M., Birkstedt, · Teemu, & Viljanen, M. (2022). Defining organizational AI governance. *AI and Ethics 2022 2:4*, *2*(4), 603–609. https://doi.org/10.1007/S43681-022-00143-X

McDonald, C. (2019, May 22). *Streaming ML Pipeline for Sentiment Analysis Using Apache APIs: Kafka, Spark, and Drill (Part 2)*. https://dzone.com/articles/streaming-machine-learning-pipeline-for-sentiment-1

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/J.ASEJ.2014.04.011

Meszaros, J., & Ho, C. hsing. (2021). AI research and data protection: Can the same rules apply for commercial and academic research under the GDPR? *Computer Law & Security Review*, *41*, 105532. https://doi.org/10.1016/J.CLSR.2021.105532

Mishra, S. K., Saini, N., Saha, S., & Bhattacharyya, P. (2022). Scientific document summarization in multi-objective clustering framework. *Applied Intelligence*, *52*(2), 1520–1543. https://doi.org/10.1007/S10489-021-02376-5/METRICS

Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. *Social Network Analysis and Mining*, *11*(1), 1–19. https://doi.org/10.1007/S13278-021-00776-6/TABLES/1

Naraine, M. L., Wear, H. T., & Whitburn, D. J. (2019). User engagement from within the Twitter community of professional sport organizations. *Https://Doi.Org/10.1080/23750472.2019.1630665*, *24*(5), 275–293. https://doi.org/10.1080/23750472.2019.1630665

Nasar, Z., Jaffry, S. W., & Malik, M. K. (2018). Information extraction from scientific articles: a survey. *Scientometrics*, *117*(3), 1931–1990. https://doi.org/10.1007/S11192-018-2921-5/METRICS

O'Mara-Eves, A., Thomas, J., McNaught, J., Miwa, M., & Ananiadou, S. (2015). Using text mining for study identification in systematic reviews: A systematic review of current approaches. *Systematic Reviews*, *4*(1), 1–22. https://doi.org/10.1186/2046-4053-4-5/TABLES/3

Poecze, F., Ebster, C., & Strauss, C. (2018). Social media metrics and sentiment analysis to evaluate the effectiveness of social media posts. *Procedia Computer Science*, *130*, 660–666. https://doi.org/10.1016/J.PROCS.2018.04.117

Ray, M. (n.d.). *Social network | Definition, Examples, & Facts | Britannica*. Retrieved July 28, 2023, from https://www.britannica.com/technology/social-network

Santos, I., Rech, L., & Moraes, R. (2022). A Topic Modeling Method for Analyzes of Short-Text Data in Social Media Networks. *EPiC Series in Computing*, *82*, 112–121. https://doi.org/10.29007/KR1Z

Sengupta, E. (2019, May 12). *TwitterAnalysis: Social Media Sentiment Analysis using Spark and Kafka*. https://github.com/eshza/TwitterAnalysis#4-results

Sentiment Analysis: Concept, Analysis and Applications | by Shashank Gupta | Towards Data Science. (n.d.). Retrieved July 31, 2023, from https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

Social media | Definition, History, Examples, & Facts | Britannica. (n.d.). Retrieved July 28, 2023, from https://www.britannica.com/topic/social-media

Stavros, C., Meng, M. D., Westberg, K., & Farrelly, F. (2014). Understanding fan motivation for interacting on social media. *Sport Management Review*, *17*(4), 455–469. https://doi.org/10.1016/J.SMR.2013.11.004

Strengthening international cooperation on AI | Brookings. (n.d.). Retrieved June 29, 2023, from https://www.brookings.edu/articles/strengthening-international-cooperation-on-ai/

Taha, W. A., & Yousif, S. A. (2023). Enhancement of text categorization results via an ensemble learning technique. *AIP Conference Proceedings*, *2457*(1). https://doi.org/10.1063/5.0122942/2872928

Taylor, J., & Pagliari, C. (2018). Mining social media data: How are research sponsors and researchers addressing the ethical challenges? *Research Ethics*, *14*(2), 1–39. https://doi.org/10.1177/1747016117738559/ASSET/IMAGES/LARGE/10.1177_1747016117738559-FIG3.JPEG

Text Mining Scientific Articles: Why You Need the Full Picture | CCC. (n.d.). Retrieved September 18, 2023, from https://www.copyright.com/blog/text-mining-scientific-articles-need-full-picture/

TLS vs. SSL: What's the Difference? - Rublon. (2023, August 16). https://rublon.com/blog/tls-vs-ssl-whats-the-difference/

Top 10 Applications of Sentiment Analysis in Business. (n.d.). Retrieved July 31, 2023, from https://www.analyticsvidhya.com/blog/2023/01/top-10-applications-of-sentiment-analysis-in-business/

Topic Analysis: A Complete Guide. (n.d.). Retrieved August 1, 2023, from https://monkeylearn.com/topic-analysis/

Topic Modeling: Algorithms, Techniques, and Application - DataScienceCentral.com. (n.d.). Retrieved August 1, 2023, from https://www.datasciencecentral.com/topic-modeling-algorithms-techniques-and-application/

TV ratings: how many people watched the 2023 Champions League Final? - AS USA. (n.d.). Retrieved September 25, 2023, from https://en.as.com/soccer/tv-ratings-how-many-people-watched-the-2023-champions-league-final-n/

UEFA Executive Committee approves new club competition | Inside UEFA | UEFA.com. (n.d.). Retrieved September 25, 2023, from https://www.uefa.com/insideuefa/about-uefa/news/024c-0e9941616a90-f26bd21de788-1000--uefa-executive-committee-approves-new-club-competition/

van Bekkum, M., & Zuiderveen Borgesius, F. (2023). Using sensitive data to prevent discrimination by artificial intelligence: Does the GDPR need a new exception? *Computer Law & Security Review*, *48*, 105770. https://doi.org/10.1016/J.CLSR.2022.105770

Vayansky, I., & Kumar, S. A. P. (2020). A review of topic modeling methods. *Information Systems*, *94*, 101582. https://doi.org/10.1016/J.IS.2020.101582

Waehner, K. (2019, October 31). *Machine Learning and Real-Time Analytics in Apache Kafka Applications*. https://www.confluent.io/blog/machine-learning-real-time-analytics-models-in-kafka-applications/

Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J., & Stein, J. (2015a). Building a replicated logging system with Apache Kafka. *Proceedings of the VLDB Endowment*, *8*(12), 1654–1655. https://doi.org/10.14778/2824032.2824063

Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J., & Stein, J. (2015b). Building a replicated logging system with Apache Kafka. *Proceedings of the VLDB Endowment*, *8*(12), 1654–1655. https://doi.org/10.14778/2824032.2824063

Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review 2022 55:7*, *55*(7), 5731–5780. https://doi.org/10.1007/S10462-022-10144-1

Weiner, J. (2021). Why AI/Data Science Projects Fail. *Why AI/Data Science Projects Fail*. https://doi.org/10.1007/978-3-031-01685-1

Westergaard, D., Stærfeldt, H. H., Tønsberg, C., Jensen, L. J., & Brunak, S. (2018). A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts. *PLOS Computational Biology*, *14*(2), e1005962. https://doi.org/10.1371/JOURNAL.PCBI.1005962

What is Apache Kafka? | IBM. (n.d.). Retrieved July 28, 2023, from https://www.ibm.com/topics/apache-kafka

What is Kafka, and How Does it Work? A Tutorial for Beginners. (n.d.). Retrieved July 28, 2023, from https://developer.confluent.io/what-is-apache-kafka/

What is Sentiment Analysis? A Complete Guide for Beginners. (n.d.). Retrieved July 31, 2023, from https://www.freecodecamp.org/news/what-is-sentiment-analysis-a-complete-guide-to-for-beginners/

What is Text Mining? | IBM. (n.d.). Retrieved September 18, 2023, from https://www.ibm.com/topics/text-mining

Wunderlich, F., & Memmert, D. (2020). Innovative Approaches in Sports Science—Lexicon-Based Sentiment Analysis as a Tool to Analyze Sports-Related Twitter Communication. *Applied Sciences 2020, Vol. 10, Page 431*, *10*(2), 431. https://doi.org/10.3390/APP10020431

Yan, X., Guo, J., Lan, Y., & Cheng, X. (2013). A biterm topic model for short texts. *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, 1445–1455. https://doi.org/10.1145/2488388.2488514

Zambrano, P., Torres, J., Tello-Oquendo, L., Yánez, Á., & Velásquez, L. (2023). On the modeling of cyber-attacks associated with social engineering: A parental control prototype. *Journal of Information Security and Applications*, *75*, 103501. https://doi.org/10.1016/J.JISA.2023.103501

Zuiderwijk, A., Chen, Y. C., & Salem, F. (2021). Implications of the use of artificial intelligence in public governance: A systematic literature review and a research agenda. *Government Information Quarterly*, *38*(3), 101577. https://doi.org/10.1016/J.GIQ.2021.101577