PHOO WAI NYEIN

Implementing Automated Testing Strategy for Legal Decision Understanding AI

DEGREE PROGRAMME IN DATA ENGINEERING
2022

Author(s)
Nyein, Phoo Wai

Type of Publication
Bachelor's thesis

Date
Month Year Jun 2023

Number of pages 43

Language of publication: English

Title of publication
Implementing Automated Testing Strategy for Legal Decision Understanding

Degree Programme
Data Engineering

Abstract

This thesis focuses on the automated testing strategies for the data stream pipeline that utilizes Optical Character Recognition (OCR) and Natural language Processing(NLP) for processing legal decisions. The work presents the implementation of an automated testing strategy that validates 23 business rules and evaluates the performance of the system as a whole. The testing approach focuses primarily on end-to-end testing systems for real-time Kafka streams . The strategy aims to ensure the overall quality of processing business rules throughout the pipeline, the reliability, efficiency, and accuracy of the system as a whole for processing legal decisions.

This paper explores the insights of how AI has been used in Legal Decision Making, combined Optical Character Recognition (OCR) and Natural Language Processing (NLP) techniques for processing business rules and automating legal decisions. It also discusses the methodology, tools, and techniques for automated testing and highlights the benefits of automated testing for data stream pipelines.

Keywords
automated decision-making(ADM), Artificial Intelligence(AI) , AI in legal industry ,Automation , Testing

Acknowledgments

I would like to express my gratitude to Ari Juntunen, the CTO and founder of Elinar Oy, for providing me with the opportunity to write my thesis. I would also like to thank my colleagues at Elinar for their support throughout this project. Additionally, I would like to extend my thanks to my supervisor Petteri for guidance and instruction during the writing process.

CONTENTS

LIST OF SYMBOLS AND TERMS

OCR Optical character recognition
NLP Natural language processing
NER Named-entity recognition

AI Artificial Intelligence
ML Machine Learning
DL Deep Learning
GUI Graphical User Interface

1 INTRODUCTION

Software testing plays a crucial role in ensuring the quality and reliability of software systems.(Marvin Zelkowitz,2006,p.179) It is the process of evaluating a system or application to identify any defects, errors, or gaps in functionality. Through testing software, we can uncover problems bugs and flaws in initial stage, thereby enhancing the overall quality and performance of the product.

We will explore the comprehensive overview of the system architecture, focusing on individual components such as OCR (Optical Character Recognition) and NLP (Natural Language Processing) within the pipeline. Various testing approaches are explored, including Kafka event-based real-time testing to ensure data is streaming throughout the pipeline and performing end-to-end testing to confirm everything is working as intended. End-to-end testing verifies the proper integration and functionality of producers, consumers, and brokers, the functionality of the pipeline, and overall system reliability. we will also discuss how to perform integration testing, functional testing, and unit testing for data stream pipelines with Kafka-based systems. This research will also uncover the process of manual testing and how it transitions to automated testing to ensure the pipeline works sufficiently and works as intended. The main goal of this project is to ensure the reliability and accuracy of the AI system when processing legal documents.

1.1 Motivation and Background

During my internship at Elinar , I was part of a team working on a project that involved processing decision PDFs using OCR scanning and NLP .I had a significant role in implementing business rules based on business rules specifications and testing them

for validating business rules. This experience allowed me to understand the importance of testing in the software development process to ensure the quality of the software. As a result, I decided to pursue this research topic for my thesis.

1.2 About Elinar Oy

Elinar Oy Ltd. is a limited company headquartered in Pori, Finland, founded in 1994. Elinar has been an IBM Business Partner since its inception and has developed a market-leading portfolio of IBM content and information management technology solutions, particularly in the area of AI-based data management. With over 27 years of experience in data and information management, Elinar has developed efficient solutions to solve complex business problems and specializes in case management methodologies. Despite being a relatively small company with around 20 employees, Elinar has a strong presence in the industry and is well-positioned to continue providing innovative solutions in the cognitive era.

1.3 Objective and Scope of the study

This thesis will cover the general idea of how the program extracts tests using OCR techniques and NLP, and how rules are processed throughout the pipeline. In addition, we will uncover insights of the project workflow and system architecture to understand how the business rules are process throughout the pipeline which strongly relies on Kafka for message delivery. The primary focus of the thesis is on implementing an automated testing strategy for data stream pipelines.

2 SYSTEM ARCHITECTURE

The system architecture for this project is designed to handle data and OCR processes using Kafka message delivery. The system requires Docker and Docker-compose to be installed on the host machine.

The system is made up of several components, each responsible for a specific task in the pipeline. These components include Incoming Rest, Datacap Interface, Layout Editor, Miner Connector, Business Rule Runner, and Msg Assembler. This pipeline architecture has been implemented and developed by Joonas Helava, a highly skilled senior developer at Elinar Oy.

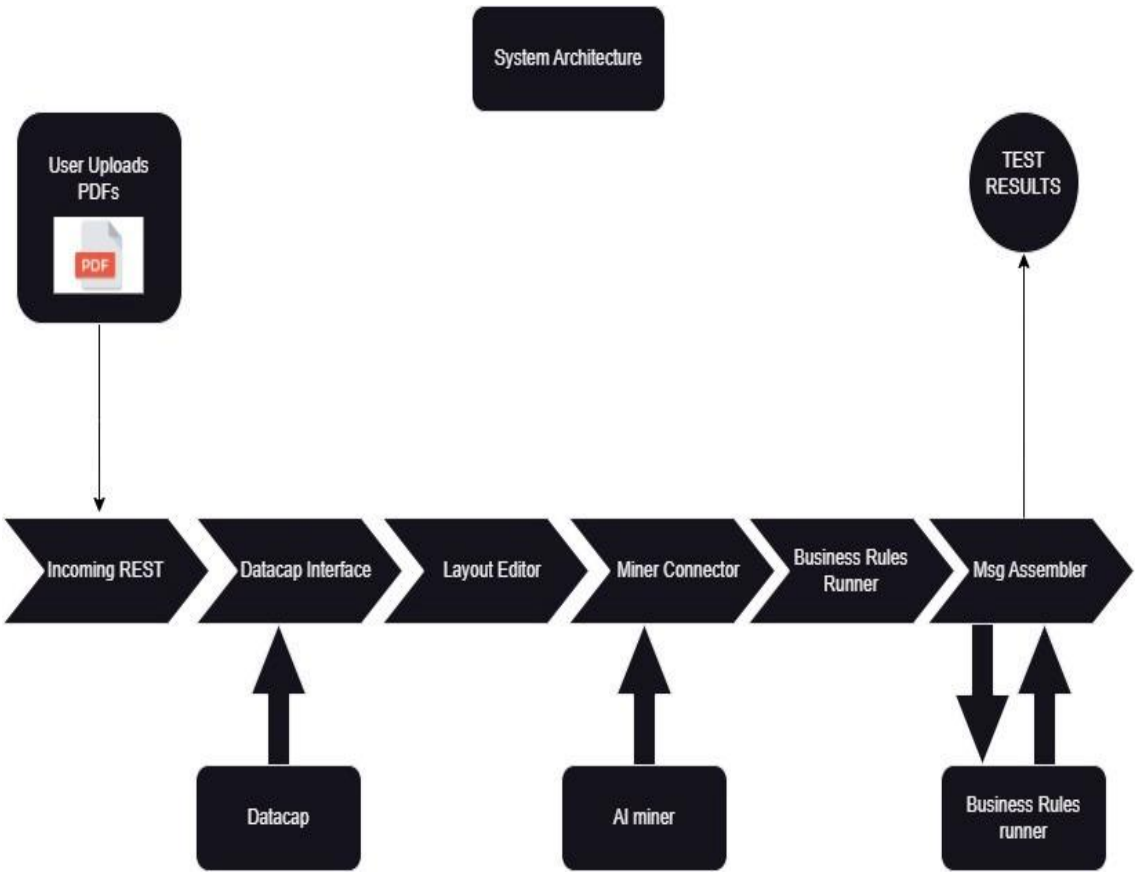

Figure 1 : System Architecture (Elinar Oy ,2020)

The system starts with a user uploading a legal decision PDF, which is then processed through the Incoming Rest, Datacap Interface, Layout Editor, Miner Connector, Business Rule Runner, and Msg Assembler components. The result is an XML

response that is returned through a REST API. The pipeline is streaming data through Apache Kafka, which is running inside a Docker container. Kafka provides a reliable and efficient in delivering messages between components, enabling seamless communication throughout the pipeline.

Overall, the system architecture is designed to handle the extraction of specific entities from legal documents using NER and OCR technologies and to provide a reliable and efficient means of processing large volumes of data.

2.1.1 Introduction to Kafka

Kafka is a popular distributed streaming platform that was originally developed by LinkedIn, and was subsequently open-sourced in early 2011. It is designed to efficiently and reliably process large streams of data in real-time. It provides a unified platform for building real-time data pipelines and streaming applications, making it a critical component in modern data architectures.

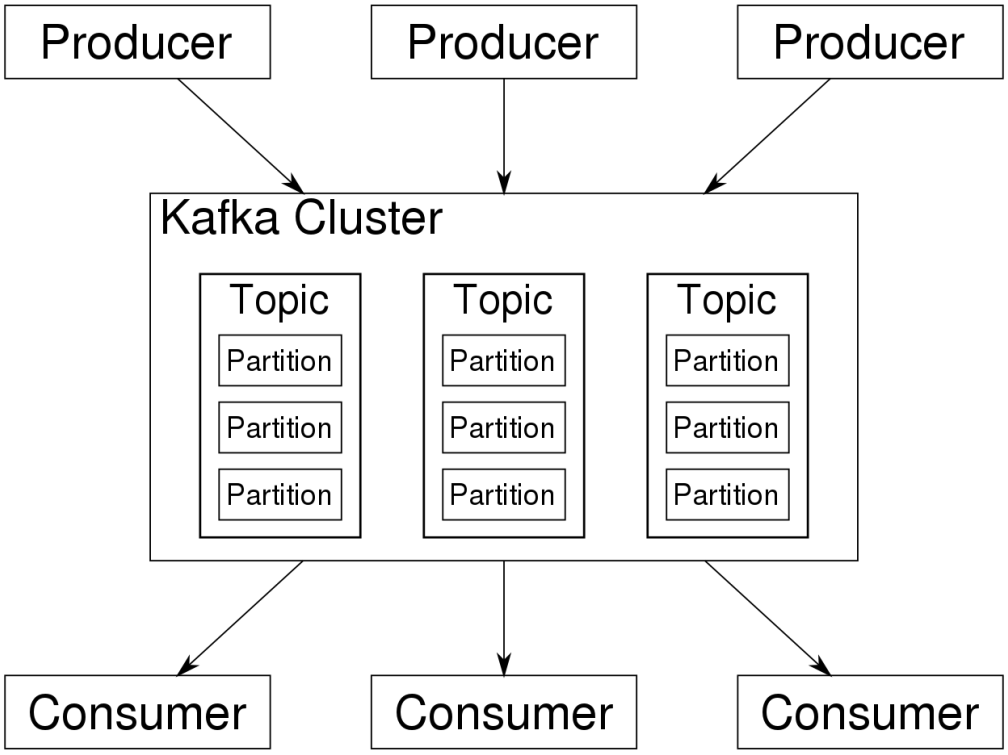

Figure 2: Apache Kafka core concept (Wikipedia,2023)

2.1.2 Kafka Streaming

At its core, Kafka enables the fault-tolerant and distributed processing and storing of continuous streams of records. It enables real-time data import, publication, and consumption, supporting event-driven systems, stream processing, and real-time analytics. Because of Kafka's flexible streaming characteristics, it may be used for a variety of purposes, including data integration, log aggregation, messaging systems, and more.(Confluent Documentation,2014-2023)

2.1.3 Kafka Producer and Consumer

Kafka Producer is a client application that publishes events to Kafka cluster. Kafka Producers are mainly responsible for writing data records to Kafka topics, which are logical categories or streams of data. When a producer sends a record, if first connects to one of the bootstrap servers which provides a list of brokers in the cluster, along with the metadata like topics, partitions, and replication factor. The producer determines the leader broker that hosts the leader partition of the producer record based on the list of brokers and metadata details and writes to the broker.(Confluent Documentation,2014-2023)

Kafka Consumer is to retrieve the data from Kafka topics by subscribing to one or more topics from assigned partitions. Consumers are mainly responsible for reading data records from Kafka Broker. It is important to note that every record is given to only one consumer within a topic. When a new consumer starts, it is assigned to a consumer group and Kafka makes sure that each partition is eaten by just one consumer from the group.(Confluent Documentation,2014-2023)

2.1.4 Kafka Topic

Kafka topics organize and exchange messages in a way that databases use tables for the same purpose. Topics are identified by a unique name and messages are held in a

logical order, making it easier to send and receive data between Kafka servers. Kafka topics mainly serve as dedicated units for organizing messages or events. When a producer sends messages to a Kafka topic, these messages are appended one after another, forming a log file. Producers have the ability to push messages to the end of these logs, while consumers can pull messages from a particular topic. Kafka topic acts as virtual groups or logs that allows users to achieve efficient communication and data management within the Kafka cluster. (Confluent Documentation,2014-2023)

In this data stream pipeline, there are several topics. Every single topic delivers data to the next topic within the pipeline and here are the Kafka topics used in the pipeline:

- ElinarAIBusinessRuleRunnerToMsAssembly
- ElinarAIPostprocessToAI
- Elinar.AI.ReceiveToOCR
- Elinar.AI.AIToBusinessRuleRunner
- ElinarAIMsgAssemblyToMsgSender
- Elinar.AI.PostprocessToAi
- ElinarAIError
- ElinarAIOCRToPostprocess
- mx
- Elinar.AI.Error
- ElinarAIReceiveToOCR
- ElinarAIAIToBusinessRuleRunnerAssembly
- Elinar.AI.MsgAssembleToMsgSender(Elinar Oy, 1994)

2.1.5 Kafka Partitions

Data streams in Kafka are split up into partitions. An offset serves as a special identifier for each record in a partition. Partitions allow data to be distributed across multiple brokers in a cluster. Kafka achieves high throughput and low latency for both data ingestion and consumption by distributing data across partitions. (Dattel, 2023)

2.1.6 Kafka Magic

Kafka Magic is a graphical user interface tool that allows users to find and display messages between topics, review and manage topics .The interface is easy to use and it offers to perform various tasks related to Kafka. Kafka magic supports JavaScript to navigate schemes and automate the task which allow users to achieve desired results. Overall, Kafka magic simplifies the complexities of real-time data processing and managing streamline.(Kafka Magic,2019)

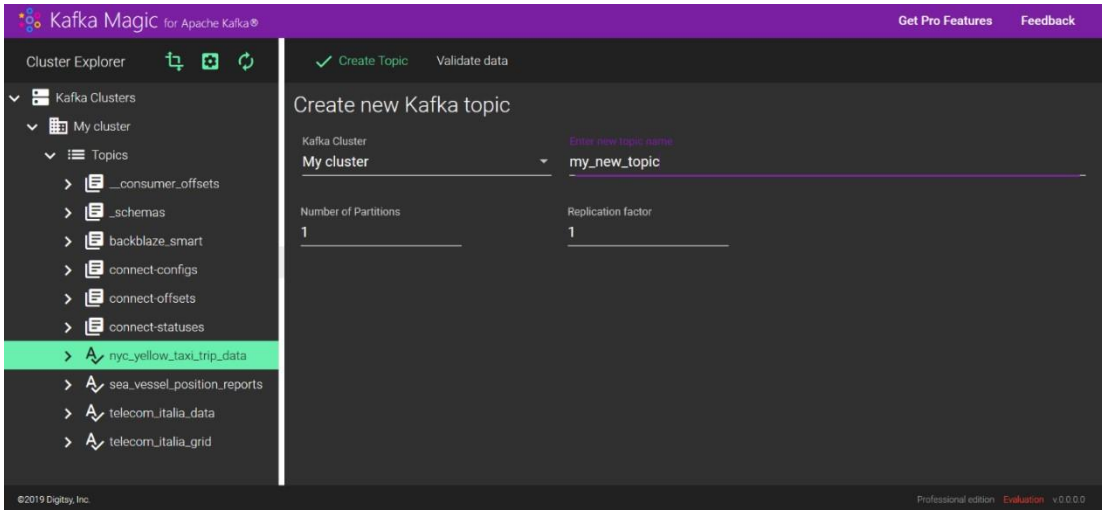

Figure 3: Screenshot of Kafka Magic GUI Tool (Kafka Magic,2019)

2.2 Optical Character Recognition Technology for Efficient Data Extraction

OCR is a technology that processes a digital image by locating and recognizing characters, such as letters, numbers, and symbols. (What is OCR,2019) OCR is a convenient tool that provides a solution for many businesses and consumers, with a significant impact on the digitization and storage of business records. The use of OCR can eliminate certain job tasks, perform tasks rapidly, and implement them at a low cost.(IBM, 2022)

OCR (Optical Character Recognition) is a process that converts the physical form of a document into a digital format. It works by scanning the document and analyzing it for light and dark areas, where the dark areas are identified as characters that need to be recognized. The scanned image is then converted into a two-color version, with the dark areas representing characters and the light areas representing the background. The

characters are then processed and recognized using algorithms that target one character, word, or block of text at a time. This involves identifying alphabetic letters and numeric digits using either pattern recognition or feature recognition algorithms.(IBM, 2020)

OCR is often used as a hidden technique for data entry automation and indexing documents for search engines, such as passports, license plates, invoices, bank statements, business cards, and automatic number plate recognition.(Wikipedia,2016)

The main benefits of OCR include low cost, accelerated workflows, automated document routing and content processing, and improved overall service by ensuring the most up-to-date and accurate information.(IBM, 2022)

2.3 What is Artificial Intelligence?

Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable. However, decades before this definition, the birth of the artificial intelligence conversation was denoted by Alan Turing's seminal work, "Computing Machinery and Intelligence" which was published in 1950.At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving. It also encompasses sub-fields of machine learning and deep learning, which are frequently mentioned in conjunction with artificial intelligence. These disciplines are comprised of AI algorithms which seek to create expert systems which make predictions or classifications based on input data. (IBM,n.d.)

2.4 An Overview of Machine Learning, Deep Learning, and Deep Transfer Learning

Deep learning(DL) and machine learning(ML) are sub-fields of artificial intelligence(AI), and deep learning is actually a sub-field of machine learning. Deep learning is comprised of neural networks. “Deep” in deep learning refers to a neural network comprised of more than three layers—which would be inclusive of the inputs and the output—can be considered a deep learning algorithm. Machine learning is more dependent on human intervention to learn. Human experts determine the hierarchy of features to understand the differences between data inputs, usually requiring more structured data to learn. (IBM,n.d.)

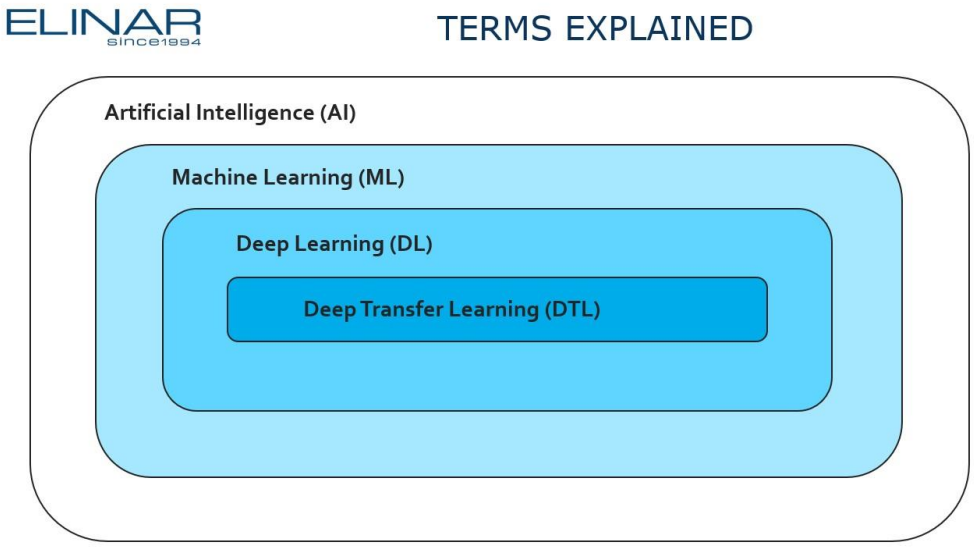

Figure 4 : Terms Explained (Ari Juntunen,2023)

Deep transfer learning uses pre-training data, which is a substantial amount of data gathered from various sources, including gigabytes or terabytes of data from the internet, domain data from back-end systems in bulk, and even specific examples like Finnish medical records. Unsupervised training is the first step in most cases, during which the neural network discovers and extracts patterns from the pre-training data without the use of annotations. Following unsupervised training, a language model is frequently constructed using the previously learned network. To make the model more specific and task-oriented, it undergoes further training using specific data related to the particular use case. In the last phase, the model is fine-tuned and adjusted in

accordance with the unique requirements of the task. The duration of this process can vary from hours to days, depending on the complexity of the task.(Ari Juntunen,2023)

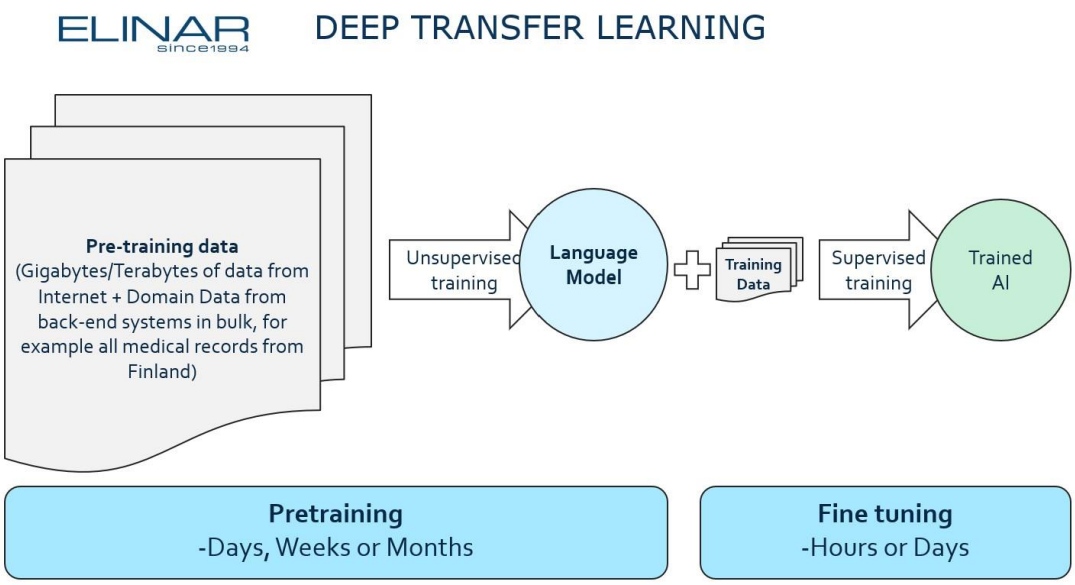

Figure 5 : Deep Transfer Learning Explained (Ari Juntunen,2023)

2.5 NLP Techniques for Data Extraction

Natural Language Processing (NLP) is a critical component in the automation of data extraction from legal documents. This project aims to use NLP techniques to extract valuable information from legal documents, such as dates, office names, and file IDs. The extracted data can then be used to improve the accuracy and efficiency of the OCR process. There are many benefits of NLP combined with OCR and one of the top-level benefits is helping businesses automate processes in real-time and we can tailor NLP tools to our needs. (Elinar Oy,1994)

The main benefits of using NLP in combination with OCR include increased efficiency, accuracy, and automation of document processing. This project aims to provide insights into how NLP can be tailored to meet the needs of businesses and organizations that deal with large amounts of legal documents. The resulting solution will provide a comprehensive and automated solution for data extraction from legal documents. (Elinar Oy, 1994)

2.5.1 Named Entity Recognition

In this project, NER is used to extract specific entities from legal documents in order to improve the OCR process. To achieve this, the team has manually tagged relevant entities such as office names, dates, file IDs, fine amounts, intentionality, issuing authorities, and more. The tagging process involves identifying the entities within the text of the legal documents and categorizing them based on their entity type, which creates a high-quality training dataset for the NER model.(Elinar Oy, 1994)

By using a quality training dataset, the NER model can accurately identify and classify relevant entities in the legal documents, which in turn can improve the OCR process and lead to more accurate data extraction.

There are multiple ways to tag entities, including tagging individual words, tagging entire sentences, or tagging intercalated words. Below are some screenshots of the tagging process that showcase how the relevant entities, such as EvidenceProvider, IssuingAuthority, fineAmount, and many more, are labeled and categorized based on their entity type.

Figure 6 : Tagging words(Elinar Oy,1994)

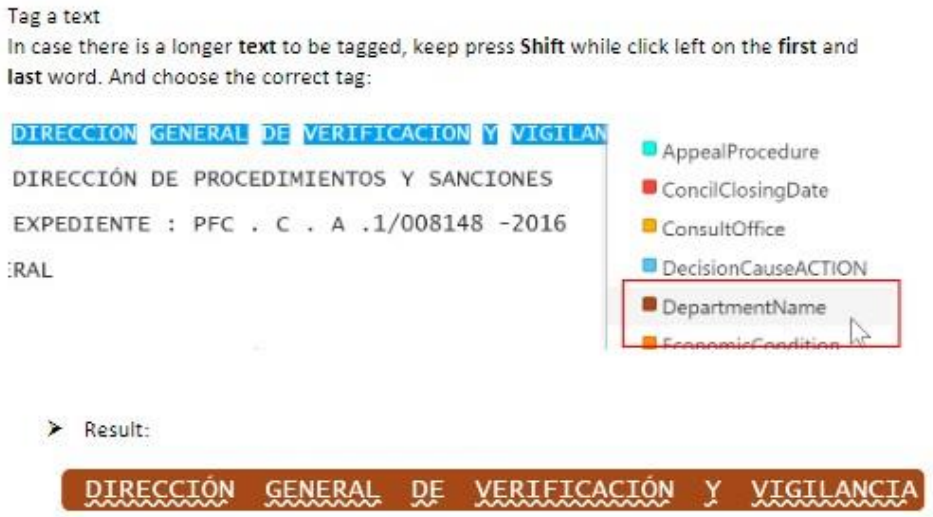

Figure 7: Tagging Texts(Elinar Oy,1994)

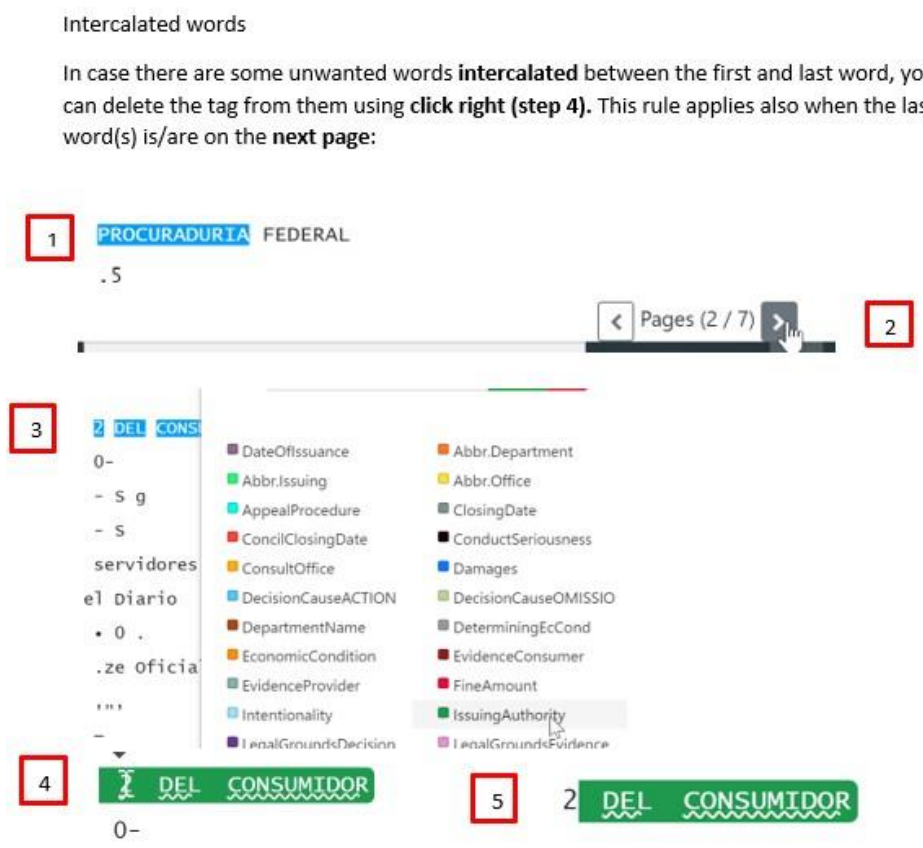

Figure 8: Tagging Intercalated words(Elinar Oy,1994)

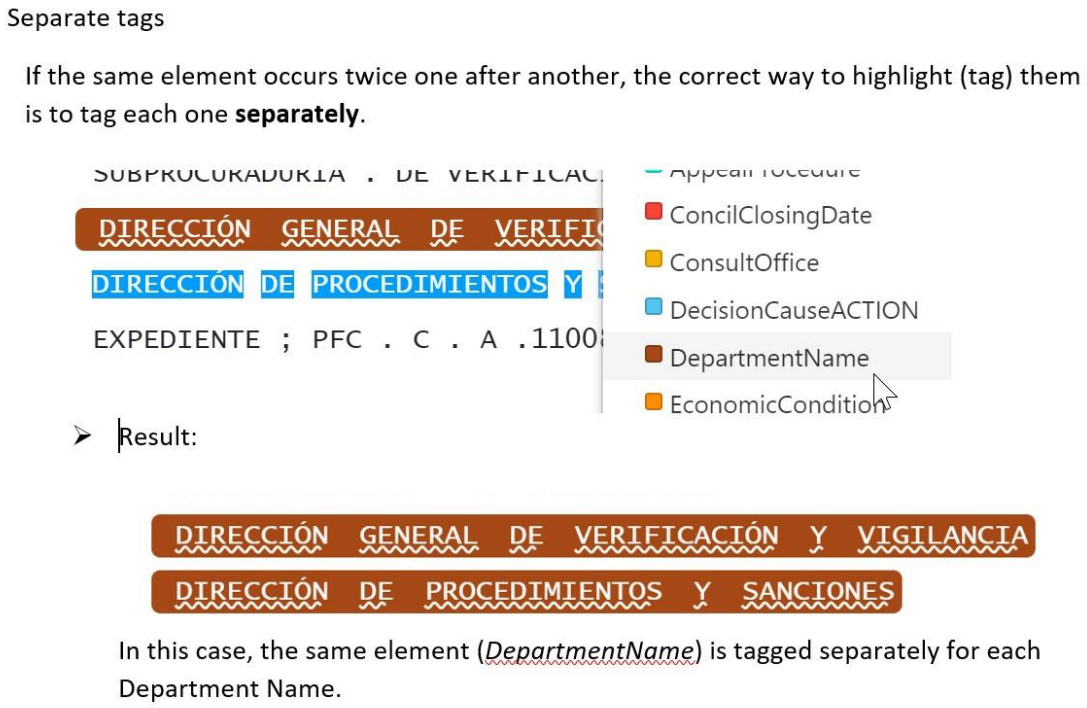

Figure 9: Separate tags(Elinar Oy, 1994)

2.6 Processing Customized Business Rules

Once the legal documents are parsed using OCR, the extracted data is passed through a series of 23 distinct business rules that has been processed with Python. Each business rule involves specific variables, parameters, and examples that guide the logic required for processing the rules.

Difficulty	Name	Variables	Parameters	Result
1	Issuing Authority error	IssuingAuthority	Indicating an Issuing Authority different than PROFECO Procuraduría Federal del Consumidor	Nullity
2	Office or Department names absence	OfficeName, DepartmentName	Absence of issuing Office and/or Department	Nullity
3	Out of time to proceed with sanctions	ConcilClosingDate, StartOfProcedure	There are more than 30 business days between the two dates	Nullity

Figure 10: Example Business Rules(Elinar Oy,1994)

In Figure 8, consider rule #1, which involves finding out if a decision is made by a different authority or not. In this rule, the Python code searches for specific keywords or authority names such as "PROFECO" and "Procuraduria Federal del Consumidor" using a rule-based approach. However, not all business rules can be implemented using a rule-based approach. Some rules require the extraction of date information from the date entity to calculate the duration associated with legal proceedings.

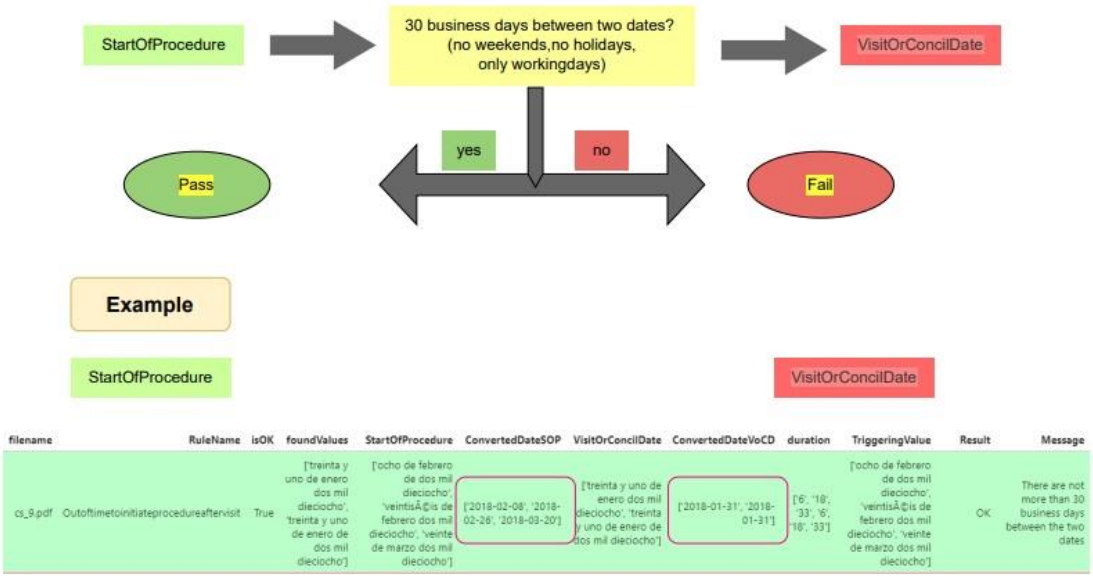

Figure 11 : Business rule logic (Elinar Oy,1994)

In Figure 11, The rule determines whether a resolution was issued within 30 business days from the closing date and date of issuance.

Figure 12 : Example Date Format in Legal Document(Elinar Oy,1994)

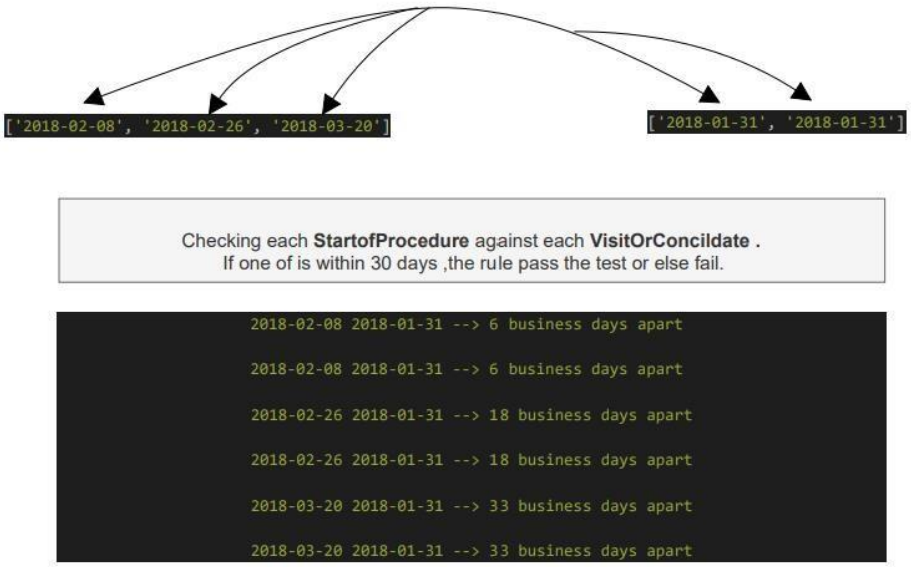

Figure 13 : Process of 30 Business Day Rule(Elinar Oy,1994)

The extracted dates are then passed through a function called date_converter, which uses regex to parse the date and convert it into the proper date format. The code then uses a specialized Mexican calendar that accounts for holidays and non-business days specific to Mexico to calculate the duration between the two dates. If the duration is less than or equal to 30 business days, the rule is considered passed. Otherwise, the rule is considered failed. This approach ensures accurate calculation of business days and the precise processing of the rule. (Elinar Oy,1994)

2.7 Regex Pattern Matching for Legal Document Processing

Regular expressions conform a powerful tool for developers that allow pattern matching which is used for manipulating text and extract data Regex allows for efficient and precise pattern matching, making it an essential tool for extracting specific data from complex documents. (Anubhava Srivastava, 2017, p.1) In this project, Regex has been used to parse dates, extract relevant clauses and provisions, and determine whether specific rules have been processed within 30 business days.

Here is a screenshot of some example regex patterns used for parsing different date formats. Note that in total, there are 23 regex patterns used in this project for date parsing.

Figure 14 : Regex Patterns for Parsing date(Elinar Oy,1994)

When parsing a legal document, OCR errors can sometimes occur frequently and repeatedly. In these cases, regular expressions (regex) can be a useful tool for accurately parsing the data despite the errors. In this project, regular expressions (regex) are used not only for extracting date but also for converting it to the proper date format.

2.7.1 Handling OCR Errors and Incomplete Date Formats

The date extraction process can be affected by OCR errors, such as misspellings or incomplete dates.

Figure 15 : Date Regex Patterns Passed and Failed for Parsing Date(Elinar Oy, 1994)

One common OCR error is with the year format, where year 2016 may be written as 'dos mil dieciséls' or 'dos mil dieciséla.' Similarly, months can be misspelled, such as 'Enero' written as 'enew,' or 'Febrero' as 'lebrero.' Additionally, the year 2000 may be written as 'Dos Mil,' which can be transcribed in various ways, such as 'dee mil' or 'das mil.'

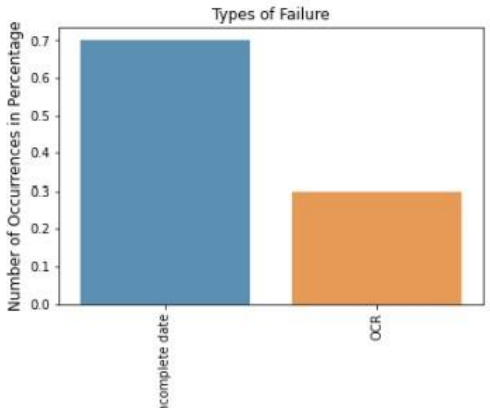

Figure 16 : Types of Failure

The date converter function is designed to handle the majority of OCR errors, with a success rate of approximately 95%. However, in some cases where the date format has multiple OCR errors, the converter may not work correctly. For instance, if the date includes multiple misspelled words or incomplete formats, the function may not be able to extract the correct date.

	concilclosingdate	converted_days	converted_years	converted_d_m_y	results	test_status
0	integer012 de abril del integer011	01 de abril del integer011	01 de abril del 2018	01 de abril del 2018	2018-04-01	Passed
1	integer012 de abril del integer011	01 de abril del integer011	01 de abril del 2018	01 de abril del 2018	2018-04-01	Passed
2	integer029 de septiembre integer011	01 de septiembre integer011	01 de septiembre 2018	01 de septiembre 2018	2018-09-01	Passed
3	integer029 de septiembre integer011	01 de septiembre integer011	01 de septiembre 2018	01 de septiembre 2018	2018-09-01	Passed
4	cuatro de julio dos mil dieciséis	cuatro de julio dos mil dieciséis	cuatro de julio dos mil dieciséis	cuatro de julio dos mil dieciséis	2016-07-04	Passed
5	cuatro de julio dos mil dieciséis	cuatro de julio dos mil dieciséis	cuatro de julio dos mil dieciséis	cuatro de julio dos mil dieciséis	2016-07-04	Passed
6	veinte de septiembre del dos mil dieciséis	veinte de septiembre del dos mil dieciséis	veinte de septiembre del dos mil dieciséis	veinte de septiembre del dos mil dieciséis	2016-09-20	Passed
7	veinte de septiembre del dos mil dieciséis	veinte de septiembre del dos mil dieciséis	veinte de septiembre del dos mil dieciséis	veinte de septiembre del dos mil dieciséis	2016-09-20	Passed
8	integer010 de septiembre de integer020	01 de septiembre de integer020	01 de septiembre de 2018	01 de septiembre de 2018	2018-09-01	Passed
9	integer010 de septiembre de integer020	01 de septiembre de integer020	01 de septiembre de 2018	01 de septiembre de 2018	2018-09-01	Passed
10	integer010 de septiembre de integer020	01 de septiembre de integer020	01 de septiembre de 2018	01 de septiembre de 2018	2018-09-01	Passed
11	integer010 de septiembre de integer020	01 de septiembre de integer020	01 de septiembre de 2018	01 de septiembre de 2018	2018-09-01	Passed

Figure 17 : Passed Parsing Date

In summary, regular expressions parser have proven to be the most effective in parsing complicated data. Phil Wilkins(2022,p.85) In this project, data parsing has been one of the complex tasks and it has been solved by defining a regex pattern that aligns with the structure of a specific date format. In general,regex is a powerful tool for extracting and converting dates from various sources and enable efficient data processing. In the present example, we can see that most dates are successfully parsed and converted to the expected format. This proves the reliability and versatility of regex patterns in dealing with complicated date formats and structures. By using regex patterns, developers can streamline data parsing tasks and improve the accuracy and consistency of data processing in the applications or data pipelines.

3 THE SIGNIFICANCE OF SOFTWARE TESTING

Testing is an integral part of the software development lifecycle. Any software needs to be tested and evaluated before the deployment phase which verifies the functionality, performance, and overall quality of the software. In the context of testing the data stream pipeline, the purpose is to ensure the reliability of the workflow throughout the pipeline, the accuracy of NLP, and the quality of OCR and functionality of the business rules validator. Testing should be performed continuously at every stage of development such as OCR updates, changes in functionality, and business rule improvements. By testing frequently and thoroughly, we can avoid the overwhelming number of bugs and errors and ensure that pipelines adhere to high-quality standards.

3.1.1 Test Goal

When developing Stream application, we end up implementing a bunch of functional component and a typical test begins with reading data from Kafka Topic .(Prashant Kumar Pandey,2019, p.247) The goal of testing is to ensure the overall quality, reliability, and performance of the software, which includes checking various aspects such as data entry, data transformation, delivery of expected results, and also whether the system is reliable and stable enough under higher workloads. In addition, testing aims to get the expected results from each component at the end of the iteration, and therefore frequent testing is essential for many aspects. This includes validating the accuracy and correctness of the data, ensuring data transformation throughout the pipeline, identifying problems or bottlenecks that may occur during the process. (Pedro Henrique,2023)Testing the system from various aspects makes it easier for developers to deal with any problems that may arise and which enables to deliver a high-quality and reliable solution that meets the requirements.(Tools QA,2022)

3.1.2 Testing Strategies

There are various types of testing for data stream pipelines. (Thoughtworks,2023) The following are the types of testing relevant to this project:

- Unit Testing
- Integration Testing
- Data Quality Testing
- Performance Testing
- End-to-End Testing

3.1.3 Unit Testing

Unit Testing involves testing a single component of the pipeline in isolation to ensure the functionality of the component. By conducting unit test, we can identify that transformations are functioning as expected and address the issue to improve the quality of the transformations. Overall, unit testing is an important step in the data processing workflow by assuring that each individual component of the code is functioning well and generating the reliable result. (Thoughtworks , 2023)

3.1.4 Integration Testing

An integration test is an important part of the software testing process, as it involves checking individual components or units of a software project to expose defects and problems to verify that the application is working as intended. (Javatpoint,2021)
Integration testing is important because it shows us how the different parts of a system work together. Each part may be fine on its own, but problems can occur when they work together. Integration testing helps uncover problems or incorrect behavior that

may occur when the units are combined. This ensures that all components of the system as a whole function correctly and work together seamlessly. By testing the integration of these units, we can identify and fix problems before they become major issues in the overall system. In general, Integration test is a primary phase of the software testing process when testing is done at the development stage.(Lambdatest,2023)

3.1.5 Data Quality Testing

Data quality testing is the practice of making assertions about your data, and then testing whether these assertions are valid. This concept can be used to test both the quality of your raw source data and to validate that the code in your data transformations is working as intended. (DBT, 2023) The purpose of testing the quality of the data is to seek the accuracy, consistency, and reliability of the data within the pipeline. Data quality testing measures the accuracy of OCR, by taking the output of scanned results and comparing it to the original version of the same document. By testing, we can ensure that the extracted text is reliable, correctly captured, and errorfree to proceed throughout the pipeline.

3.1.6 Performance Testing

Performance testing is a test that evaluates the speed, reliability, and stability of the application. The purpose of performance testing is to avoid poor user experience. By conducting performance tests, we can determine if the application meets desirable speeds and responsiveness while under workloads.(Thoughtworks,2023) For instance, we may measure the response time of the pipeline when a user uploads a single document and multiple documents. These files goes through the entire workflow and ultimately return an XML response. The focus is on understanding how the application performs under different scenarios.

Another aspect of performance testing to compare how the system handles larger file sizes comparing toe smaller ones .By doing that, we can determine if there are any

significant differences in performance .Overall, performance testing is to ensure that data pipeline can handle the expected workload efficiently and reliably.

3.1.7 End-to-end testing

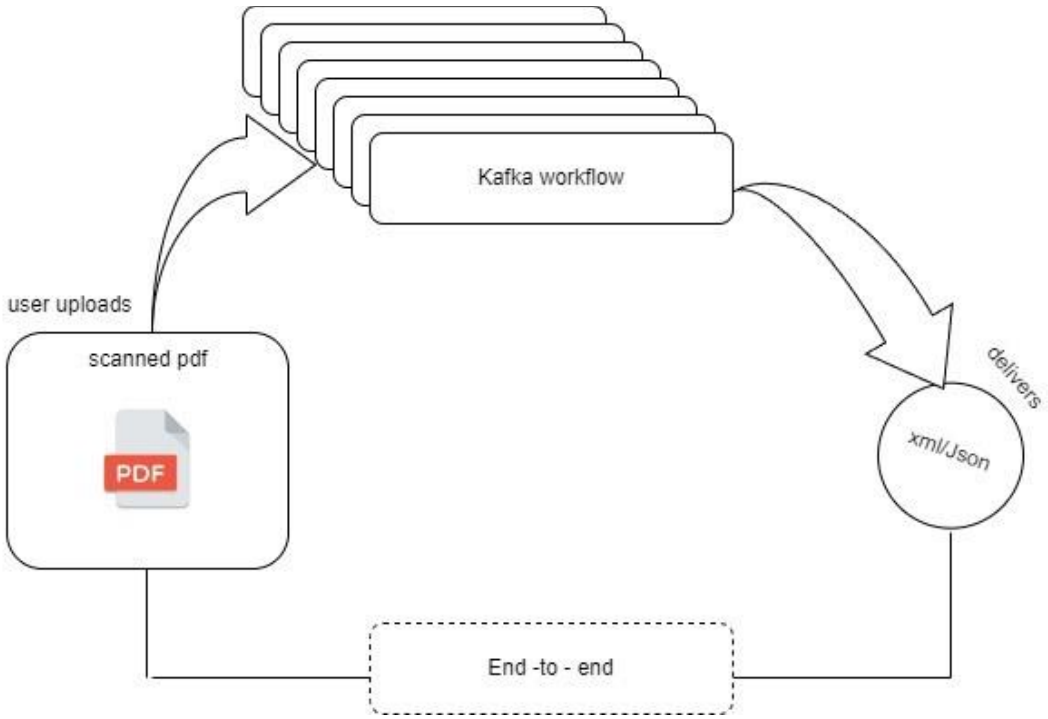

Figure 18 : End-To-End Testing (Elinar Oy, 1994)

End-to-end testing is a comprehensive technique that accesses the complete work flow of the application, ensuring that all components work together and the software functions correctly in real world scenario.

In End-to-end testing, the software is tested from the perspective of an end-user by simulating user viewpoint. The primary objective is to verify the behavior of the program as a whole, including its functionality, dependability, performance and security.E2E testing aids in finding flaws such as server failure, file size limitations and the availability of handling invalid documents that could arise when several components of the application interact with one another. It also verifies that all parts of the pipeline function flawlessly together to meet user expectations by testing the complete system.

End-to-end gives the following advantages:

- It makes sure that all components function properly together as a whole and match the necessary business objectives.

- It spots the flaws before they become more complicated and expensive, allowing developers to fix them in the early stage.

- It lowers costs related to errors and faults found late in the development process or after the program is launched.

- It ensures that the application satisfies the needs of the business, increasing the likelihood that users will adopt it

- It helps streamline the testing process by testing the application from user's perspective rather than individual components in isolation.

In summary, end-to-end testing is a crucial role in verifying component compatibility, finding flaws early on, and fulfilling business requirements. End-to-end testing helps companies provide high-quality software that satisfies client expectations and reduces potential problems by extensively testing the entire data stream. (BrowserStack, 2023)

3.2 REST API Testing with Postman

API stands for Application Programming Interface which allows programs to interact with each other and exchange data between them. (Juliusz L. Kulikowski, 2011)APIs define how developers should request services from other programs, resulting in higher productivity and revenue. API can be implemented using different protocols such as SOAP and REST. SOAP is based on XML and focuses on function-driven communication. In contrast, REST, which is based on HTML, structures data using

XML, YAML, or JSON and is more data-driven. Several types of requests can be used when making requests to an API, including GET, POST, PUT, PATCH, and DELETE.(Hevo, 2023)

In this project, the primary method for interacting with messaging system is “POST” method. Kafka Producers use this method to post messages to Kafka topics by sending them to Kafka brokers. The project has set up a router for handling HTTP POST requests to the server. It handles three different types of POST requests, including saving a file to the server, collecting a file from the server, and polling for the existence of a file on the server.

The process of saving a file to the server is handled by the /save endpoint. It then saves the request body to a server file after determining whether the request is coming from an authorized hostname.

The /poll/ endpoint is in charge of determining whether a file is present on the server at the end. The server responds to the client with true or false depending on whether the file is there or not.

A file is obtained from the server using the /collect/ endpoint. It then reads the file indicated by the file Id parameter after performing an authentication check using the Basic Auth scheme.

3.2.1 Test Implementation with Postman

To initiate our first request with Postman, we will need to provide a valid username and password to login endpoint URL . Once the request is sent, we should receive a 200 status code as a successful response along with the authentication token which can also be used for subsequent authenticated requests.

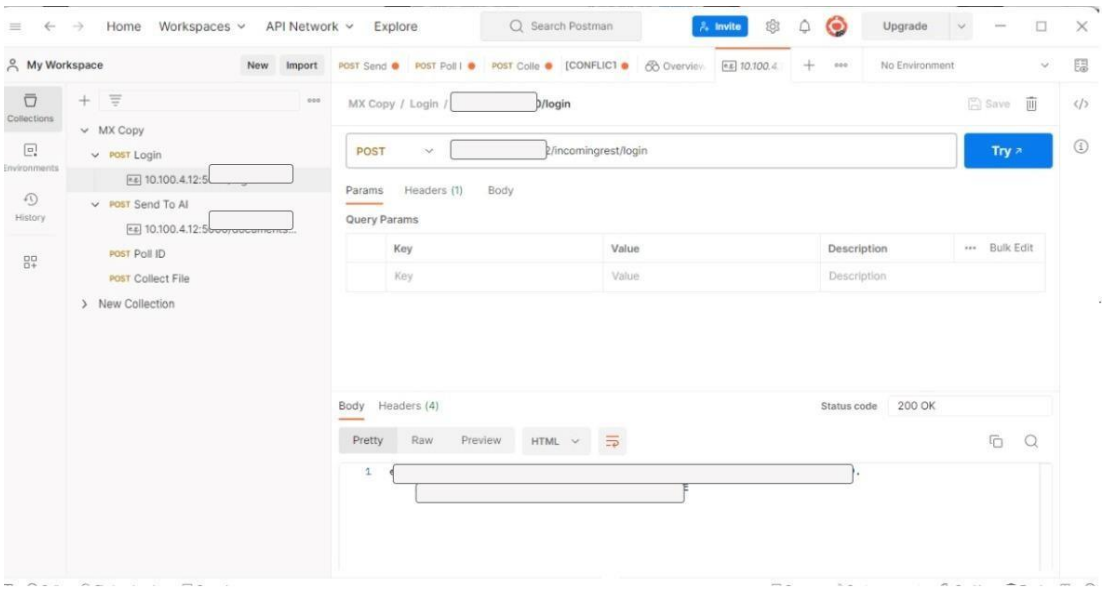

Figure 19 : Postman Login Authentication

After successfully logging in, we are good to proceed with sending file to the server. We can select a decision PDF of our choice in body section, attaching an authorization token in the header section to upload the file to the "sendtoai" endpoint URL. Upon successful upload, we should receive a 201status code which indicates that the send request is completed and the file is uploaded. Additionally, we will also receive a JSON response containing the relevant file ID associated with the uploaded file.

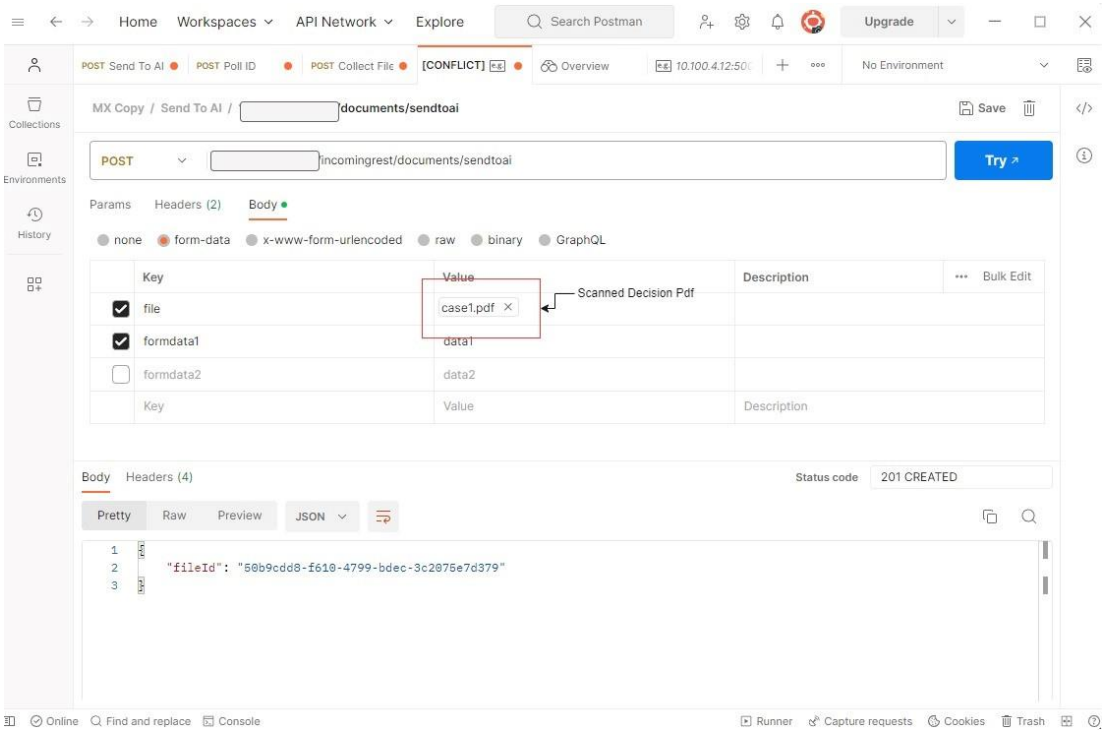

Figure 20 : Sending Document with Postman (Elinar Oy, 1994)

Once the file is successfully uploaded, the uploaded file will process throughout the pipeline which involves several components. This process takes some time to complete. Therefore, we can send a poll request to the poll endpoint URL to check if the response is ready to be collected or not. We will continue to poll until we receive a response of “True” which indicates that the process is complete, and it is ready to be collected from collect endpoint URL.

3.2.2 From Postman to Python for Automated Testing

Although Postman is commonly used for API testing, it has limitations in flexibility, such as a limit on the amount of documents that can be processed at one time. This research describes automated testing strategies that save time and effort by eliminating repetitive tasks. In addition, automating the testing process provides us with several benefits in terms of efficiency, accuracy, and time savings. Through automation, we can reduce manual effort, eliminate human error, and ensure consistency. Automated testing also allows us to complete the test faster and get feedback on the quality of the system.

Setup:

In this section, we walk through the end-to-end automation process. We start by installing the requests library using the command "pip install requests." The requests library enables easy sending of HTTP requests and remains one of the most widely used Python packages.(Real Python, 2012)

Code Implementation:

Following the same steps we took with Postman, we now replicate the process using the Python request library. First, we log in and then send the file to the server.

```
Import requests
 def sendToAI(auth_token, file):     try:                 filename
= os.path.basename(file)         hed = {'path':'<folder name>/',
'destination': <destination>,
'Authorization': <authorization token>}
with open(file, 'rb') as f:
            files = {'file': f}             response =
requests.post(myurl, headers=heders, files=files,
verify=False)         fileId = response.text
                return fileId     except
requests.exceptions.HTTPError as errh:
        return "Failed request :", errh     except
requests.exceptions.ConnectionError as errc:
        return "Failed request :", errc
except requests.exceptions.Timeout as errt:
        return "Failed request :", errt     except
requests.exceptions.RequestException as err:
        return "Failed request :", err
```

Program 1 : Sending Document with Request (Elinar Oy, 1994)

Result:

After we send the file, we should receive the same response from server as we received in Postman, including the file ID, which is associated with the file. This confirms that the file was successfully sent to the server using the Python library.

The next step is to continuously poll the result using Python's while loop until we get a True response. When we get a True response, we may collect data from collect endpoint and receive an XML response in our terminal that looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<env:DocumentEnvelope
xmlns:env="http://www.elinar.com/document/envelope/v1">
<env:Header>
        <env:Process>AI Process</env:Process>
        <env:Id>1ab5489a-4054-42fd-9b07-c3e0be277d30</env:Id>
        <env:ReferenceId>1ab5489a-4054-42fd-
9b07c3e0be277d30</env:ReferenceId>
    </env:Header>
    <env:OriginalContent>
        <env:DocumentType>Document</env:DocumentType>
        <env:Type>Binary</env:Type>
        <env:MimeType>application/pdf</env:MimeType>
        <env:Content></env:Content>
    </env:OriginalContent>
    <env:Attachments>
        <env:Attachment>
            <env:DocumentType>Elinar AI results</env:DocumentType>
            <env:Sender/>
            <env:FileName/>

<env:ContentDefinitionReference>Elinar.AI.Results.v1</env:C
ontentDefinitionReference>
            <env:MimeType>application/xml</env:MimeType>
            <env:Content><![CDATA[<Elinar.AI.Results>
<ResponseHeader>
<ProcessingTime>2023-04-22 06:08:02</ProcessingTime>
<ProcessingError>
<ErrorCode/>
<ErrorMessage/>
</ProcessingError>
</ResponseHeader>
<Document>
  <DocumentHeader>
    <PageNumbers>1,2,3,4,5</PageNumbers>
    <TotalPages>5</TotalPages>
    <DocumentParts>5</DocumentParts>
    <PartNumber>1</PartNumber>
    <PartType>Page</PartType>
    <DocumentLossInfo>
      <MaxLoss>1.0</MaxLoss>
      <AvgLoss>1.0</AvgLoss>
      <MinLoss>1.0</MinLoss>
    </DocumentLossInfo>
  </DocumentHeader>
  <elements/>
  <BusinessRules>
   <!--- Business Rules validation results goes here-->
</BusinessRules>
```

```
</Document>
</Elinar.AI.Results>]]></env:Content>
        </env:Attachment>
    </env:Attachments>
</env:DocumentEnvelope>
```

The responses are wrapped around with CDATA sections to escape and preserve special characters within an element, which allows the inclusion of characters that would otherwise be treated as markup or parsed by XML parser. CDATA sections are typically used when the content of an element contains characters such as angle brackets (< >) or quotation marks (“) that need be preserved as literal text .

We can store the response xml files in a local folder, to proceed with the test results we can parse XML using Etree module of the lxml library. To begin, we import the require modules and create an XML parser with specific options such as removing CDATA sections and not resolving entities.

```
             from
lxml import etree
from io import StringIO, BytesIO
  parser = etree.XMLParser(strip_cdata=True,
resolve_entities=False) doc = etree.parse("example.xml", parser)
root = doc.getroot() ns = root.nsmap attachments =
root.find("env:Attachments", ns) attachment =
attachments.find("env:Attachment", ns) content =
attachment.find("env:Content", ns) contentxml =
etree.parse(StringIO(content.text), parser) document =
contentxml.find("Document")
businessrules = document.find("BusinessRules")
 for businessrule in
businessrules:
    # Code within the loop to handle each business rule
# ...
```

Program 2 : Reading business rules test results from xml (Elinar Oy, 1994)

The code snippet shows how the business rules validation results has been parse from the XML response which allows further procession with generating test report .(Elinar Oy, 1994)

Finally, to automate the whole process, we can combine all the above steps with Python code. This includes logging in, sending the file, retrieving the XML response, retrieving the business rules response, and generating a report showing which rules work and which do not. By implementing these steps in a cohesive Python script, we achieve end-to-end automation. This approach streamlines the testing process, saves time and effort, and ensures a more efficient and reliable testing process.

3.2.3 Why is end-to-end automated testing crucial for this project, and what are the challenges?

Building and maintaining high-quality apps requires automated end-to-end testing. However, it has drawbacks that can get in the way when applications get more complicated and have more interrelated parts. Testing such applications comes with its own set of difficulties, particularly when many capabilities operate in various ways. A modification is more likely to mistakenly affect unrelated areas the more complex an application's functionality is. The more extensive the functionality of an application, the higher the likelihood that any modification may unintentionally impact unrelated areas. Both developers and testers need to ensure seamless interaction among all the components in their applications. Manual testing is typically performed for this purpose, but it can be tedious and prone to mistakes. One of the most effective ways to ensure a positive user experience with your applications is through automated endto-end testing, which, alongside other automated tests like unit and API testing, is a vital part of the modern software development lifecycle.(Telerik,2021)

End-to-end testing plays a crucial role in creating high-quality software applications. It adds to the effectiveness of other types of testing like functional or unit testing by expanding the scope of testing and working the system as a whole. In applications where several services interact with one another, an end-to-end test can identify problems that may not be revealed by other isolated tests. End-to-end tests offer

thorough coverage but managing them has become more difficult as a result. These tests can be sluggish and unstable, necessitating extra care to assure their execution. Here are the top 3 challenges for end-to-end testing :

1. Test Flakiness

End-to-end testing often suffers from a problem known as test flakiness. These tests have a reputation for being unstable and unreliable, causing them to fail unpredictably. It can be incredibly frustrating when your test suite passes smoothly one day, only to encounter unexplained failures the next. The situation becomes even more aggravating when you rerun the tests, and everything miraculously passes without any issues.(Telerik,2021)

The complexity of end-to-end tests, which involve multiple components and validate various aspects of the system, makes it challenging to pinpoint the cause of failures. This inconsistency leads to a significant loss of productivity as teams struggle to identify the root cause of the flakiness. (Telerik,2021)

2. Slow Test

In addition to flakiness of the tests, there is also criticism of the slowness of end-toend testing. As mentioned earlier, end-to-end tests go through the entire stack of your application and examine every component, both internal and external. It would be unfair to compare the testing time of these tests to the fast execution of unit or functional tests that validate only specific parts of your application.(Telerik,2021)

3. Long-Term Maintenance

End-to-end tests are known for their extensive coverage of different scenarios, involving multiple steps and assertions compared to other types of tests. For example,

in this project, the test begins with the user sending a file to the server to validate business rules. The system then proceeds to extract data, utilize NLP to extract entities, and interacts with other components that implement specific business processes. As a result, the flow covers a significant amount of ground.

The more end-to-end test cases we accumulate, the larger the test suite becomes and the more comprehensive it becomes. Over time, it becomes a challenge for testers to manage these tests effectively. It becomes difficult to add new tests without affecting the existing ones, and even small changes to the application under test can cause the entire test suite to fail. That's why it's so important to develop your end-to-end test suite with long-term maintenance in mind.(Telerik,2021)

4 CONCLUSION

Two types of automated testing can be performed for this project: end-to-end testing with rest and Kafka consumer testing. Rest API testing involves validating the functionality of business rules implemented in Python throughout the entire pipeline. This type of testing ensures that the system is working correctly and that the business rules are being applied as intended. Rest API testing is performed by sending HTTP requests to the API endpoints and verifying the response data, status codes, and headers returned by the server. Automated Rest API testing is faster and more reliable than manual testing and helps to catch issues early in the development process.

Kafka consumer tests are run to see if messages are delivered as intended or not. The main purpose of Kafka consumer-producer tests is to evaluate the effectiveness of Kakfa data streams. We can get the pipeline's functionality and the results of the business rules validation by running consumer tests. It is advised to automate Kafka consumer-producer tests since it takes into account a number of factors, including the speed, dependability, and scalability of Kafka data streams.

In summary, end-to-end testing covers the overall system behaviour, including Kafka interactions, whereas Kafka producer and consumer testing focuses specifically on testing the functionality and performance of the producer and consumer components in a Kafka-based system. Both types of testing are important in ensuring the reliability and quality of a distributed system.

REFERENCES

Elinar Oy (1994) https://www.elinar.com/

Marvin Zelkowitz.(2006).Adbances in Computers : Web Technology- page 179 : https://www.google.fi/books/edition/Advances_in_Computers/dG0Dlc6YGmMC?hl=en&gbpv=1&dq=Software+testing+plays+a+crucial+role+in+ensuring+the+quality+and+reliability+of+software+systems.&pg=PA179&printsec=frontcover

Apache Kafka Concept Diagram(2023,May 21).[Photograph].Retrieved from Wikipedia.https://en.wikipedia.org/wiki/Apache_Kafka

Confluent. (2023, May 24). Kafka Streams Documentation. Retrieved from Confluent Documentation. https://docs.confluent.io/platform/current/streams/concepts.html

Dattell. (n.d.). What is a Kafka Partition? Retrieved from Dattell Data Architecture Blog: https://dattell.com/data-architecture-blog/what-is-a-kafka-partition/

Screenshot of the Kafka Magic tool ().Retrieved May 24, 2023, from https://www.kafkamagic.com/

Optical character recognition. (2016, April 15).In Wikipedia.Retrieved May 24, 2023, from https://en.wikipedia.org/wiki/Optical_character_recognition

NECC.(2019).What-is-OCR?.Retrieved from https://www.necc.mass.edu/wpcontent/uploads/accessible-media-necc/uncategorized/resources/What-is-OCR.pdf

Optical character recognition (OCR) technology(2022, January 5).IBM cloud Education,Retrieved May 24, 2023,from https://www.ibm.com/cloud/blog/opticalcharacter-recognition

Anubhava Srivastava(2017) .Java 9 Regular Expressions.Packt Publishing

:https://www.google.fi/books/edition/Java_9_Regular_Expressions/1eZDDwAAQBAJ?hl=en&gbpv=0

Phil Wilkins(2022).Logging in Action. page 85: https://www.manning.com/books/logging-in-action

Prashant Kumar Pandey(2019).Kafka Streams - Real-time Stream Processing.Learning Journal.page 247 : https://www.google.fi/books/edition/Kafka_Streams_Real_time_Stream_Processin/ms6NDwAAQBAJ?hl=en&gbpv=0

Pedro Henrique(2023).Data Quality 101: Ensuring accurate data in your pipelines.https://medium.com/@pedro.schleder/data-quality-101-ensuring-accuratedata-in-your-pipelines-f1649c2c5241

Test Goal.(2022, January 16).Tools QA. Why is Testing necessary and Important?, from https://www.toolsqa.com/software-testing/istqb/why-is-testing-necessary/

JavaTpoint. (n.d.)Integration Testing.Retrieved May 27, from javaTpoint: https://www.javatpoint.com/integration-testing

Lambdatest.(n.d.)Integration Testing Tutorial: A Comprehensive Guide With Examples And Best Practices.Retrieved May 27, from : https://www.lambdatest.com/learning-hub/integration-testing

DBT.(n.d.) Data quality testing: Where and why you should have it.Retrieved May 27, from : https://www.getdbt.com/blog/data-quality-testing/

BrowserStack(2023,February 20).What is End To End Testing?. Retrieved May 27, from :https://www.browserstack.com/guide/end-to-end-testing

Juliusz L. Kulikowski(2011). Human-Computer Systems Interaction: Backgrounds and Applications 2.page 203 : https://www.google.fi/books/edition/Human_Computer_Systems_Interaction_Backg/UeQ8kR6lxXMC?hl=en&gbpv=0

Hevo.(n.d.)How to Work With Postman REST Client: The Complete Guide.Retrieved May 28, from : https://hevodata.com/learn/postman-rest-client/

Real Python .(n.d.)Python's Requests Library (Guide).Retrieved May 28, from :https://realpython.com/python-requests/

Telerik .(n.d.) Top challenges of automated end-to-end testing .Retrieved June 16, from : https://www.telerik.com/blogs/top-challenges-automated-end-to-end-testing