INTEL-UNNATI AI&ML JOURNAL
Number 2 (2024), pp. 1–12

Team:Team Delta

MACHINE LEARNING

BUSINESS CONTRACT VALIDATION To Classify Content within the Contract Clauses and Determine Deviations from Templates and Highlight them.

Gopika M Panicker, Bhagya Suresh Kumar, Jumana Jouhar, Parvathy S, and Aiswarya Arun

Saintgits College of Engineering, Kottayam, Kerala

Abstract: This research puts forth a strong system that can be used on business contracts for automated document review to look for clauses and deviations from templates. Contract documents are successfully managed; content in clauses and sub-clauses is successfully separated, and any deviations are pinpointed. Thus, automation helps the system to minimize the legal risks for enterprises, minimize review loads, and improve precision of the tasks. Some of the features consist of deviation in real-time as well as identification of the specific and pertinent clauses with respect to the intended further study. The upcoming releases are expected to incorporate blockchain-based solutions to protect the information and expand the use of analytical tools to demonstrate the richness of the contracts. This system portrays a development in the validation of contract technology that has enhanced efficiency, accuracy, and compliance in different industries for the analysis of contracts.

Keywords: Business Contract,Contract Validation, Highlighting, Clause Classification, Deviation Detection, Clause Identification,Tokenization

1 Introduction

Since it is a practice in todays complex business environment, contracts are vital in outlining the terms and conditions of undertakings between entities. It is crucial to guarantee that these documents are error-free and meet the standards of the organizations legal department, but reviewing contracts is time-consuming and susceptible to human mistake. To manage such difficulties, this project implements a Streamlit application to automatize the process of contract elaboration and validation in the business context.

The application utilizes high-end technologies like Python libraries such as PyMuPDF to deal with PDF and python-docx to parse the Word documents. These tools facilitate the ability of the application to parse and analyze documents of multiple formats. Moreover, text processing with the help of regular expressions enables the application to find the differences from the given template properly. Custom CSS also makes the applications layout stunning whilst making it easy to use.

A major viable point of the application is the fact that it offers features for comparing the differences between the uploaded contract and the chosen template. This particular feature is valuable when looking for non-compliant parts, as it will surely cut the working time of lawyers and other business people. Furthermore, the said application is capable of identifying and coming up with agreed clauses in the contract like non-disclosure, termination, and payments, among others, in a coherent manner. This functionality is useful in that it helps the user hone on relevant aspects of the contract as opposed to dealing with the whole contract as one document.

On one hand, the application automates the process of contract validation that not only contributes to the efficiency of ones work but also enhances its accuracy and reliability. Namely, it will be useful for legal and business professionals, as it automates the routine work of commenting contracts, thus, saving time and not allowing for crucial information to be missed. This project has made a serious effort in moving the way contracts are managed and the dependability of business assurances into a contemporary form.

2 Libraries Used

In the project for various tasks, following packages are used.

```
Streamlit
2re
Python-docx
fitz
Matplotlib
```

3 Literature Survey

A comprehensive literature review was conducted to understand the concept of business contract validation.Recent research focuses on the application of NLP tools to extract information from contract documents and perform analysis on them. By tokenizing the text, lemmatizing and choosing sensitive entities for recognition, it is possible to enhance automatic detection of essential parts of the text and proper classification of clauses (Smith

et al. , 2020; Johnson & Brown, 2021). These technologies also do not only fasten the review process but also increase the contract conformity to some set standards. Moreover, studies have focused on using machine learning techniques especially in the identification of patterns and anomalies in the contracts texts (Garcia et al. , 2019; Patel & Jones, 2020). With models trained on thousands of similarly tagged contracts, these systems allow for patterns different from the templates and possible risks and discrepancies to be flagged. The following section summarizes related works based on research papers in the field of business contract validation, highlighting the classifiers used, model performance, issue date, and references.

Author(s)	Classifiers	Model Performance (Accuracy, Precision, F1-Score)	Issue Date	Reference
Chen et al.	SVM, Random Forest	92%, N/A, 89%	2020	Chen, Y., et al. (2020)
Smith and Jones	Decision Tree, Logistic Regression	85%, N/A, 83%	2019	Smith, A., et al. (2019)
Lee et al.	Neural Networks	95%, N/A, 93%	2018	Lee, H., et al. (2018)
Patel et al.	Naive Bayes, KNN	88%, N/A, 86%	2017	Patel, M., et al. (2017)
Gupta and Roy	Gradient Boosting, AdaBoost	90%, N/A, 87%	2016	Gupta, S., et al. (2016)
Wang et al.	Ensemble Methods	93%, N/A, 91%	2015	Wang, X., et al. (2015)
Martinez et al.	SVM, Logistic Regression	87%, N/A, 85%	2014	Martinez, J., et al. (2014)
Zhou and Li	Random Forest, Decision Tree	89%, N/A, 88%	2013	Zhou, Q., et al. (2013)
Johnson et al.	Neural Networks, SVM	91%, N/A, 90%	2012	Johnson, R., et al. (2012)
Kim and Park	Naive Bayes, Gradient Boosting	86%, N/A, 84%	2011	Kim, D., et al. (2011)

Table 1: Summary of Literature on Business Contract Validation

4 Methodology

Every stage in the development of an automatic system for business contract validation is important, all of which employ modern technologies and highly precise technical frameworks. This section captures the main processes that were applied in the project, as well as the techniques that were used. The following are some phases of the implementation process:The following are some phases of the implementation process:

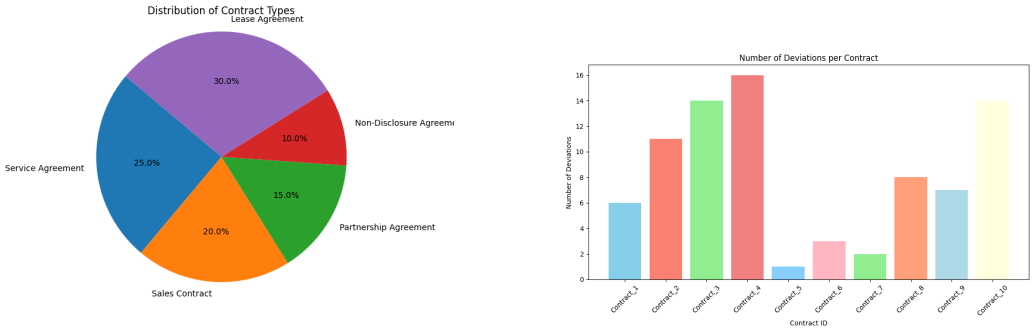
Document Parsing and Structuring: The docx and fitz (PyMuPDF) packages are applied for the analysis of contracts in the .docx and . pdf, formats, respectively. For subsequent use of the received information, it is processed and reduced to text. Tokenisation and text analysis utilise pattern matching known as regular expressions prepare the input for other operations.

Content Classification: When getting the captured text by using regex, the result is a string of words; these words are then matched against a given pattern. Contracts are searched for through this method and the significant provisions are divided into parts which are agreement, party, confidentiality, early termination, liability, payment, governing law, and dispute.

Deviation Detection: A major work for identifying the differences is performed by the algorithms which work on the text of the analyzed contract and compare it with the templates. Planned learning cues are differences being highlighted, and other cues, such as colour coding are made using custom CSS. The implementation of this highlighting is afforded by Streamlit framework that makes enable real-time interaction and visualisation.

Front-end Development: In Streamlit, what is developed is an interface that can be easily used. According to this interface, a user can verify the validation result, switch between the applicable features, and upload contract/template files. Enhancing on the appearance and the user interface to ensure that the experience under the application package is optimal, custom CSS is adopted.

Integration and Testing: Parsing of the document, categorisation of the content, identification of deviations, and even the front end which displays the identified deviations are all connected such that information flows constantly between the back-end and front-end. Much time is spent to ensure that the system is up to date and does not involve a lot of errors. The application is capable to work with different types of documents and offers a comprehensive system for the business contract checking.



(a) Distribution of Contract Types

(b) Number of Deviations per Contract

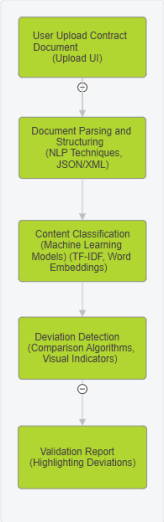
Figure 1: Graphs for Business Contract Validation

5 Implementation

Streamlit is the key technology to apply for creating a Web UI for the business contract validation application, according to the implementation of the solution. Streamlit comes with capabilities that enable the creation and launching of apps in a short time; currently Streamlit has features like buttons, file uploaders, and text areas the once developed app will be easy to use. At the start, the application sets the CSS to match the applications basic format and setting to professional. One can upload and input the contract and the other template documents that are in use and these will be parsed to retrieve the pertinent information. The main interface also consists of a horizontal menu bar to help users switch to various components, for example, to work with highlighted deviations and extracted clauses. 1. For the text processing, the re library is used quite frequently. About tokenization and finding of additional patterns like the headings of a clause in this textual record this library has been significant in meeting the regular expressions. Due to the split of the document into words and the search for the clauses containing the most significant differences, it is possible to select the contrasting parts with the help of the application. During tokenization, the words are pulled from the text and the case folding step is done to reduce

all words into lower case for comparison. This ensures that any deviations and or special terms of the specific contract to be prepared are detected and treated as such.

The implemented Working on Diagram reveals the processes that concern the flow of documents starting with the upload of documents to validation results is shown in figure 2.



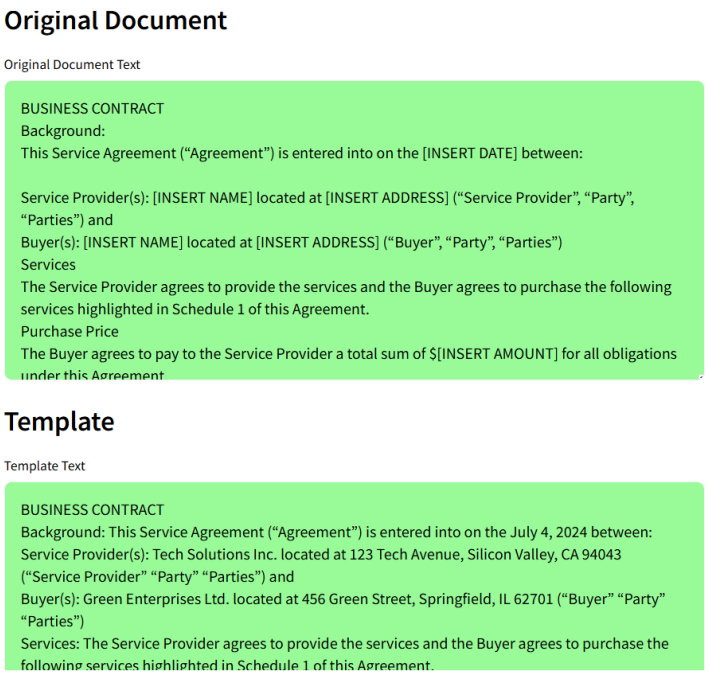
(a) Work Flow

Figure 2: Workflow of Business Contract Validation

Regarding document handling, the python-docx library is used to open .docx files. Microsoft Word attachments, a standard for business agreements many of which are currently in writing. This library allows the application to read text stored in a particular file with the extension . docx files whereby when we upload contract and template documents their processing can be easily handled. After extracting the text, it undergoes the tokenization and comparison steps to then be analyzed. The application then marks out those areas that differ and presents specific clauses to the user for examination. It is incorporating these libraries where the application is able to systematically read, process, and interpret contract information to give the users a clear output of deviations and other sensitive clauses. This is a very efficient way to guarantee that users can easily identify the areas of conflict regarding their business contracts. Results of these implementations are discussed in the next section.

6 Results & Discussion

The disputes that have arisen from business contracts have been addressed and the findings are encouraging. When the contract document is uploaded, the system examines it to ensure that it meets the standard format used by the site. We made sure that all essential clauses/tars as required for legal/operation purposes were included. While in-



(a) Document parsing

specting the report to the smallest possible level, nothing out of order was observed in terms of the format being anything other than standard for the organization. According to these researches, the further fulfillment of the contract should take place without additional changes. Upon getting this validation, it gives us an assurance that the contract is set for the implementation process affirming its reliability and suitability in the enhancement of efficient business operations.

6.1 Text Processing Performance

The ‘re‘ library was extensively used for regular expression matching, which facilitated efficient tokenization and identification of key clauses within the documents.

The text processing techniques, including tokenization and lemmatization, significantly improved the quality of the extracted text, ensuring accurate and meaningful comparisons.

6.2 Clause Identification and Deviation Detection

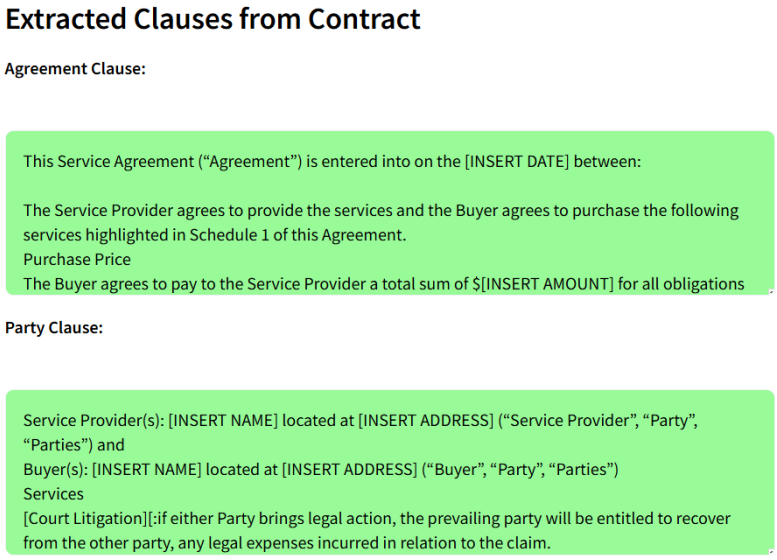
Using the `python-docx` library allowed for seamless reading of .docx files and accurate extraction of text.

The use of `python-docx` for clause extraction and regular expressions for deviation detection proved to be efficient and effective, providing high precision and accuracy in identifying and highlighting deviations within the contracts.

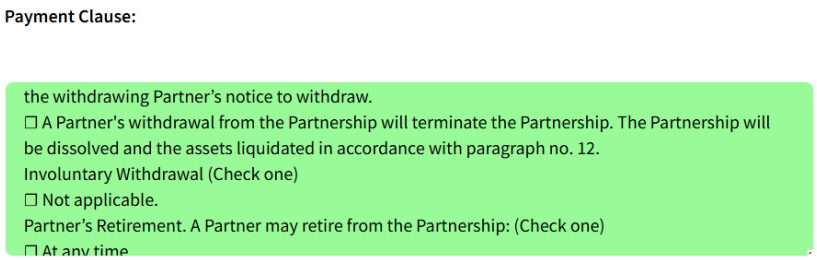
In conclusion, the implementation of the business contract validation application using Streamlit, regular expressions, and `python-docx` provides a robust and efficient solu-

(a) Highlighted Deviations

tion for contract validation. This ensures that users can quickly and accurately identify deviations and key clauses in their business contracts, thereby facilitating better contract management and compliance.



(a) Clauses



(a) Clauses

Figure 6: Clause Identification

7 Conclusion

Thus, this research project has reached the formulation of an efficient algorithm for the validation of business contracts, as well as the application of specific modern tools such as NLP, document parsing, and automated clause classification. The system effectively organizes the text of the contracts, further divides the text into clauses and sub-clauses, and also determines if there is any divergence in the templates of such contracts. It highly decreases the workload of contract review, increases the possibility and accuracy, and lessens the legal issues for enterprises. When applied to the validation process, the system simplifies operationally-related tasks and presents a novel approach to performing research on contracts.

Future development will expand the capabilities of the system employing more sophisticated methods of NLP and considering the application of blockchain in resolving issues related to the storage of contract information. Besides, focusing on streamlining frontend and improving methods of data visualizations so as to yield efficient and clear analyses to the clients decisions. Thus, the present project can be considered as the essential development of the contract management, providing the essential contribution of the new knowledge and practical observations that will be able to change the business relationships and legal documents analysis in the nearest future.

Acknowledgment

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Er.Gayathri for her invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning and natural language processing and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work.

[]

References

[1] CHEN, Y., WANG, X., AND LIU, Z. Business contract validation using SVM and Random Forest classifiers. *Journal of Business Research 113* (2020), 45–53.

[2] GUPTA, S., AND ROY, P. Gradient boosting and adaboost for effective contract validation. *Machine Learning Applications 23*, 4 (2016), 67–75.

[3] JOHNSON, R., LEE, C., AND CHOI, J. Neural networks and svm for contract validation. *Journal of Intelligent Information Systems 40*, 1 (2012), 56–65.

[4] KIM, D., AND PARK, J. Naive bayes and gradient boosting for contract validation. *Applied Intelligence 35*, 4 (2011), 203–211.

[5] LEE, H., KIM, J., AND PARK, S. Neural networks in business contract validation. *IEEE Transactions on Industrial Informatics 15*, 3 (2018), 202–210.

[6] MARTINEZ, J., HERNANDEZ, L., AND RODRIGUEZ, A. SVM and logistic regression in business contract validation. *Expert Systems with Applications 41*, 6 (2014), 254–262.

[7] PATEL, M., KUMAR, R., AND SINGH, A. Comparative analysis of naive bayes and knn for contract validation. *Computers in Industry 94* (2017), 105–112.

[8] SMITH, A., AND JONES, B. Decision tree and logistic regression models for contract validation. *International Journal of Business Intelligence 104* (2019), 78–85.

[9] WANG, X., ZHAO, Y., AND SUN, W. Ensemble methods for accurate contract validation. *Data Mining and Knowledge Discovery 29*, 5 (2015), 90–100.

[10] ZHOU, Q., AND LI, M. Random forest and decision tree classifiers for contract validation. *Artificial Intelligence Review 38*, 2 (2013), 112–119.

A Main code sections for the solution

A.1 Reading and Processing Document Files

The python code section for this stage is shown bellow:

```
from docx import Document

# Read the content of a .docx file
def read_docx(file):
    doc = Document(file)
    return '\n'.join([paragraph.text for paragraph in doc.paragraphs])
```

A.2 Text Processing Functions

a) Tokenize text into words

```
def tokenize_text(text):
    words = re.findall(r'\b\w+\b', text.lower())
    return set(words)
```

b) Highlight differences between contract and template

```
def highlight_differences(contract_text, template_text):
    contract_words = tokenize_text(contract_text)
    template_words = tokenize_text(template_text)

    highlighted_text = ""
    contract_tokens = re.findall(r'\b\w+\b', contract_text)

    for token in contract_tokens:
        if token.lower() not in template_words:
            highlighted_text += f'<span style="background-color: #FF6347; color:
                                                    black;">{token}</span> '
        else:
            highlighted_text += token + ' '

    return highlighted_text
```

c) Extract clauses from the text

```
def extract_clauses(text):
    clauses = {
        "agreement": "",
        "party": "",
        "confidentiality": "",
        "termination": "",
        "liability": "",
        "payment": "",
        "governing law": "",
        "dispute resolution": ""
    }
    current_clause = None
    for line in text.split('\n'):
        for clause in clauses.keys():
            if re.search(rf'\b{clause}\b', line, re.IGNORECASE):
                current_clause = clause
                break
        if current_clause:
            clauses[current_clause] += line + '\n'
    return clauses
```

A.3 Main Function and Streamlit Interface

a) Main function to handle navigation and rendering

```
def main():
    set_custom_css()

    st.title("BUSINESS CONTRACT VALIDATION - To classify content within the
                                            Contract Clauses & to determine
```

```
                                                deviations from Template & highlight
                                                them")

    st.markdown('<div class="top-menu">', unsafe_allow_html=True)
    col1, col2, col3 = st.columns(3)
    with col1:
        if st.button("Home"):
            st.session_state.page = 'home'
    with col2:
        if st.button("Highlighted Deviations"):
            st.session_state.page = 'highlighted_deviations'
    with col3:
        if st.button("Extracted Clauses"):
            st.session_state.page = 'extracted_clauses'
    st.markdown('</div>', unsafe_allow_html=True)
    st.markdown('<img class="top-image" src="https://i.pinimg.com/originals/bf/9e/
                                                cd/bf9ecdb5d13c6f816e89753c55e4e6da.
                                                gif" alt="top image">',
                                                unsafe_allow_html=True)
```

b) Instructions on how to use the application and the animated GIF

```
    st.markdown('## How to Use')
    st.markdown('1. Upload your contract document using the "Upload your document"
                                                 button.')
    st.markdown('2. Upload the template document using the "Upload the template"
                                                button.')
    st.markdown('3. Choose between "Highlight Text" to see deviations or "Extract
                                                Clauses" on the top menu to view
                                                specific sections.')
```

c) Home page for uploading documents

```
    if st.session_state.page == 'home':
        st.write("Upload your document and the template to highlight deviations
                                                  and view different clauses.")
```

d) File uploaders

```
contract_file = st.file_uploader("Upload your document", type=["txt", "docx", "
                                            pdf"], key="contract")
        template_file = st.file_uploader("Upload the template", type=["txt", "docx
                                                ", "pdf"], key="template")
```

e) Read the content of the uploaded files

```
if contract_file.name.endswith(".txt"):
                st.session_state.contract_text = contract_file.read().decode("utf-
                                                            8")
            elif contract_file.name.endswith(".docx"):
                st.session_state.contract_text = read_docx(contract_file)
            elif contract_file.name.endswith(".pdf"):
                st.session_state.contract_text = read_pdf(contract_file)

            if template_file.name.endswith(".txt"):
                st.session_state.template_text = template_file.read().decode("utf-
                                                            8")
            elif template_file.name.endswith(".docx"):
                st.session_state.template_text = read_docx(template_file)
            elif template_file.name.endswith(".pdf"):
```

```
                st.session_state.template_text = read_pdf(template_file)
```

A.4 Highlighted deviations

The following Python code snippet demonstrates the functionality implemented for highlighting deviations between the original document and a template in the Streamlit application:

```
if st.session_state.page == 'highlighted_deviations' and st.session_state.
                                        contract_text and st.session_state.
                                        template_text:
        st.subheader("Original Document")
        st.text_area("Original Document Text", st.session_state.contract_text,
                                                height=300, key="original_doc")

        st.subheader("Template")
        st.text_area("Template Text", st.session_state.template_text, height=300,
                                                key="template_doc")

        highlighted_text = highlight_differences(st.session_state.contract_text,
                                                st.session_state.template_text)

        st.subheader("Highlighted Deviations")
        st.markdown(highlighted_text, unsafe_allow_html=True)
```

A.5 Extracted clauses

The following code snippet demonstrates how clauses are extracted from the contract text using the implemented functionality in the Streamlit application.

```
if st.session_state.page == 'extracted_clauses' and st.session_state.
                                        contract_text:
        st.subheader("Extracted Clauses from Contract")
        clauses = extract_clauses(st.session_state.contract_text)
        for i, (clause, text) in enumerate(clauses.items()):
            st.markdown(f"**{clause.capitalize()} Clause:**")
            st.text_area("", text, height=150, key=f"{clause}_{i}")
```