**PS-17 Business-Contract-Validation**

To classify content within the Contract Clauses & to determine deviations from Template & highlight them

**Abstract**

This paper presents a completely automated system for business contract comparisons by applying natural language processing and machine learning algorithms. It is an efficient and accurate system for comparing business contracts in today's dynamic environment. We process PDF documents with advanced text extraction methods, which involves complex legal language and structure. Extracted content undergoes sophisticated preprocessing to normalize the text and ready it for analysis. At the core of our treatment lies the implementation of TF-IDF vectorization, together with cosine similarity calculations, so that on this very basis, the system is intelligent enough to suitably match similar paragraphs between the two versions of a contract. Our system has the capability to model these matched paragraphs eloquently, showing subtle differences and visually representing changes. Apart from this, it locates and identifies unique content in each document. No addition or deletion goes unnoticed in this comprehensive approach. This is the power brought to legal professionals and business stakeholders alike by streamlining this process. By greatly reducing the time needed for manual comparison and minimizing the risk of human error, our system is bound to set a new standard of confidence and accuracy in the revision of contracts. It designs, implements, and evaluates the methodology of such a system, which has been proven to have very good efficacy in real-world scenarios, holding great potential to eventually revolutionize current contract management practices.

**1. Introduction**

Modern business is a complex landscape wherein every contract undergoes several dozen revisions before its finalization. The traditional process of manual comparison among various versions is time-consuming, full of errors, and hence highly challenging for legal professionals and businesses. This paper, thus, focuses on the emerging need for an efficient and accurate method of contract comparison.

We present an automated system using NLP and machine learning to compare two versions of a contract and highlight the changes and similarities therein. Our approach transforms what was essentially a laborious manual process into a quick, accurate, and reliable automated operation. The new system exploits state-of-the-art computational techniques to do something that humans cannot: analyze quickly and comprehensively the complex language structure and structured nature of legal documents.

The technical underpinnings of our system are explored in detail in this paper, discussing issues unique to processing legal documents and illustrating how our approach overcomes such obstacles with case studies and performance metrics.

**2. Related Work**

The paper comparison has been enjoying an evolution, so to speak, with developments in the field of computer science and NLP. Eugene W. Myers already made a very crucial text comparison algorithm back in 1986, which set up an important milestone in that research area. Then, techniques for measuring semantic similarity between texts were developed, as in Mihalcea et al. It was in 2006 that Corley and Strapparava first proposed methods for semantically comparing documents. However, all these techniques fail to prove satisfactory when applied to legal documents, which need a more customized solution due to specific structure and terminology.

Such foundations, thought of great value, have inefficiencies regarding the comparison of contracts. Our approach will combine efficient techniques in matching, Semantic Understanding, and domain-specific real-world knowledge into a powerful tool for the automated comparison of contracts.

**3. Methodology**

**3.1 Model Training and Creation:**

**3.1.1 Text Extraction:**

Herein, we have used PyMuPDF (fitz) to extract text from PDF documents. The library retains the original formatting and structure.

**3.1.2 Text Preprocessing:**

* Case conversion
* Removing punctuation and special characters
* Tokenization
* Removing stop words
* Lemmatization using spacy

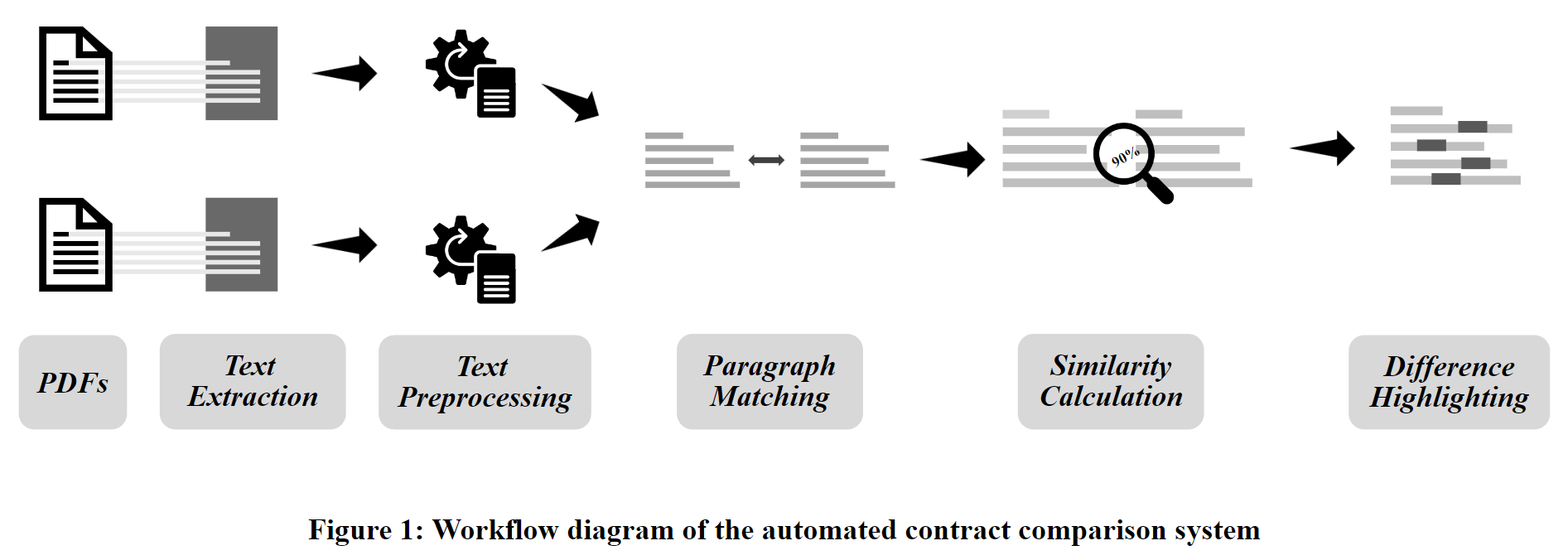
**3.1.3 Vectorization and Model Training:**

* Further segmentation is done using preprocessed text at the paragraph level. Then, it is vectorized by using TF-IDF.
* These paragraphs are matched using cosine similarity, which hence detects similarities between portions of the contracts.
* The model is trained on these vectors to identify the differences.

**3.1.4 Model Saving and Loading**

The model and a TF-IDF vectorizer are saved using joblib, and at the time of deployment, it loads the model with a vectorizer to be used accordingly.

**3.2 Contract Comparison Implementation**

****

**3.2.1 Collecting PDFs**

Our system collects from the user PDF documents that contain the business contracts to be compared. The user can upload using our interface the original and revised versions of the contracts.

**3.2.2 Text Extraction**

It exploits the PyMuPDF library for extracting text from the uploaded PDFs. That approach is presumably to maximally allow for the original format or structure of the documents to be retained, which is very important for their comparison at the paragraph level.

**3.2.3 Text Preprocessing**

The extracted text undergoes several preprocessing steps:

* **Conversion to Lowercase**: This standardizes the text by changing all characters to lowercase.
* **Removal of Punctuation and Special Characters**: Removal of punctuation and special characters cleans the text of alphanumeric characters.
* **Tokenization**: Splits the text into individual words or tokens.
* **Removal of Stopwords**: Eliminates common words (e.g., "and," "the," "is") that do not contribute to the meaning.
* **Lemmatization**: Reduces words to their base form using spaCy, improving the accuracy of the comparison by normalizing variations of words.

This process normalizes the text, reducing noise and improving the accuracy of subsequent comparison steps.

**3.2.4 Paragraph Matching Algorithm**

At this stage, the divided text is broken down into paragraphs. For this, we first extract paragraphs using regular expressions, and that helps in correctly identifying the boundaries of the paragraph. We then represent each and every paragraph into a numerical vector with the help of a TF-IDF vectorizer. Finally, the similar paragraphs in both the contracts are matched through cosine similarity. This approach also detects slight differences in words.

**3.2.5 Computation of Similarity**

Now, a score of similarity will be computed for all the matched pairs of paragraphs based on cosine similarity. Those paragraphs whose extent of similarity comes above the threshold set—that is, 0.6 for the time being—shall be characterized as matched. Of course, this threshold may be changed to bring in greater sensitivity into the comparison, if so desired.

**3.2.6 Highlighting the Dissimilarity**

We then use Python's difflib library to find word-level differences that exist within matched paragraphs. These are, in turn, highlighted in the final output to let the user know exactly what has changed from the original to the revised contract. In this manner, change visualization guards against missing any modification.

**3.3 Training the Machine Learning Model:**

**3.3.1 Model Selection and Setup:**

Before delving into the process of classification of legal clauses, the issue was solved with the help of a targeted ensemble learning approach. The ensemble comprised three distinct classifiers: The three models used here are MLP Classifier, Logistic Regression, and SVC for Support Vector Classifier. This selection was done based on the fact that this type of models provide different types of solutions for different types of problems; Neural networks for non-linearity (MLPClassifier), probability aspect (LogisticRegression) and to separate classes effectively (SVC). Applying ensemble learning on these models, the objective was to enhance the predictability and at the same time work on a diversity of legal clause types.

**3.3.2 Data Preprocessing and Feature Extraction:**

Before model training began a number of data preprocessing steps were performed to initial data sets to make them of higher quality and relevance. The raw text data of legal clauses that was obtained went through several preprocessing steps. First, the noise reduction was applied where all the characters in the text were converted to lowercase in order to obtain the required text corpus. Any [special] character other than the letters of the alphabet and textual content was stripped out pending the development’s textual content analysis. Furthermore, the known stop words like ‘and,’ ‘or,’ ‘the’ were excluded to increase the number of significance of the remaining words.

**3.3.3 Model Training and Evaluation:**

The next step was data preprocessing after which the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization feature was used. This method brought the processed text into numerical features where each legal clause’s importance was accounted for the entire corpus. In other words, the TF-IDF vectors were used also as the input features while training the ensemble model.

In the training phase, the datasets were split into the training and validation sets randomly applying a method of staggered random sampling. This was intended to keep proper representation of each class distribution in the set as split into training and validation. The chosen ensemble model was then trained on the training set with the hyperparameters tuned using a cross-validation method to prevent cases of overfitting. Since this is a binary classification problem, common large scale metrics like, Precision, Recall, and F1-Score were used to measure the performance of the model of the validation set.

**3.3.4 Training Results:**

Upon completion of training and validation, the ensemble model demonstrated robust performance across multiple evaluation metrics:

**Accuracy:**

Thus, the ensemble model reached approximately 80% accuracy score on the validation set. This metric measures how efficiently the model will be able to place each legal clause into its right category.

**Precision and Recall:**

Specificity, admits the proportion of accurately classified positive instances over all the instances classified as positive Perfectness measures the proportion of accurately classified positive instances over the total positive instances. The model of noun phrases utilized in the work provided sensible recall and precision ratios throughout various clause kinds, therefore demonstrating the machine’s capacity to discern between classes properly.

**Training Time:**

The training process was done for a time of roughly 13 min on a basic CPU. This period helped to fine-tune the parameters of the ensembles and the time-consuming computations of large data sets.

**3.3.5 Training the Model with Legal-Clause-Dataset**

For the legal clause extraction task, we employed a Legal-Clause-Dataset including 350+ different types of standard legal clauses and 150000 examples. This type of data enriched the input of the model and allowed it to focus on learning about variations that are inherent to legal clause text.

Here's a breakdown of the training process:

**Data Preprocessing:**

The text of the dataset was only converted to lowercase as well as all punctuations and stop words from the same were removed to cut down on the noise as it was focused on the semantic features of the text as mentioned earlier.

**Feature Extraction:**

Preprocessing was done on the text data and then TF-IDF vectorization was done on it. This technique converted the text into numerical aspects and went ahead and identified the significance of the words within a clause and compared it to the whole corpus. These options were used as predictors as the data input for the ensemble model.

**Model Training:**

The fully processed and already vectorized dataset was then divided into the training and validation set via the stratified manner to have an equal distribution in those sets.

Here, MLP Classifier, Logistic Regression, and SVC are the models from the ensemble model that trained on the training set.

Cross-validation was also used in the selection of hyperparameters while regularization was used in order to minimize cases of overfitting. This method involves the following steps; splitting the training data into subsets, each subset is used to train the model while the other part is used to test performance of the model.

Thus, the model performance was evaluated on the validation set using such indicators as accuracy, precision, recall, and F1-score.

**4. Implementation**

**Backend**

The Backend of the system would be implemented using Python under the Flask framework. Flask being a very simple and flexible web framework is chosen to handle a variety of backend processes: text extraction, preprocessing, matching of paragraphs, calculating similarities, and differences highlighting.

Key Libraries Used:

* **scikit-learn:** This package is developed for TF-IDF vectorization and the computation of cosine similarity. Scikit-learn is a very powerful machine learning and data analysis library that contains efficient tools for text vectorization and similarity measurement.
* **NLTK (Natural Language Toolkit) and spaCy:** NLTK has a number of utilities to clean and tokenize text; on the other hand, spaCy would be used for lemmatization in order to reduce words to their base form.
* **PyMuPDF (fitz):** Extracts text from PDF documents. PyMuPDF does an excellent job keeping the original format and structure of the documents intact for the purposes of this comparison.

**Frontend**

Next.js was the choice for the frontend due to its being a great and very popular React framework, touting server-side rendering for improved performance and SEO. This would provide a responsive and interactive user interface that offers uploading of PDF documents by users, initiation of comparisons, and finally, display of the results.

The major features of the frontend would be the following:

* **User Interface:** It would be user-friendly, intuitive to use. A simple file upload interface is provided wherein users can upload versions of the contracts: original and revised.
* **Backend Communication:** Sends the uploaded PDFs to the backend for processing, retrieves comparison results.
* **Result Display:** The result will be clear and friendly to the user. The changes in the contracts are highlighted, thus making one to easily identify the changes.

**5. Results and Discussion**

Preliminary tests conducted on the system return very encouraging results on the accuracy in the identification of similar paragraphs and on differences between contracts. It treats the great majority of standard business contracts, with the processing times dependent on document length and complexity.

Nevertheless, TF-IDF vectorization combined with cosine similarity worked quite satisfactorily in matching paragraphs, even in cases of minor changes or reorderings in the texts. It would be weaker with more heavily revised documents in which the paragraph structures had changed significantly.

Setting the threshold of similarity to 0.3 is high enough to avoid most false positives, yet low enough to catch subtle changes, but still has space for future tuning to achieve optimal performance on different types of contracts.

**6. Limitation**

However, several issues that might be completed or partially have been noted and these are in a recurrent standpoint of opportunities as well as challenges.

**6.1. Extraction of Text from PDF Documents**

About the specific approaches used in our work, one of the main difficulties we experienced during the project was the text extraction from PDF that should also maintain the formatting. In this vein, it is essential to understand that legal documents include highly complex layouts, tables, and even stylistic elements that are significant for the evaluation of legal documents. Pre-existing approaches such as PyMuPDF, can retrieve text and often fail to represent all elements like the page column, text boxes and non-linear text layout. Mentioned challenges can affect the quality of the extracted text and, accordingly, introduce deviations in the subsequent analysis and comparison.

On the positive side, there is an opportunity to enhance the text extraction accuracy with the help of LLM known as Language Model Fine-tuning, but it comes with new difficulties. Training the fine-tuning models is computationally expensive and needs annotated data for identifying the fields. Further, it is also challenging to fine-tune the model so that it captures the specifics of the legal documents’ formatting without over-learning the training dataset. In addition, implementing LLM into current systems may require level changes and changes in computation complex to support these enhancements or additions.

**6.2. Scalability and Dataset Size**

The proposed legal clause classification system, which uses the ensemble model, mainly depends on the training set size and the set’s range of variety.

**6.2.1 Impact of Dataset Size:**

The difference in the size of the dataset of the learning samples of the ensemble model is directly proportional to the overall stability and the ability of the method to generalize on other samples. More data means that there is more material for the AI model to work with: legal clauses as language units could further differ in their frequency, subtleties, patterns, and applications in legal contexts. This exposure is important because it allows the model to gather and store numerous semantic patterns and relations of the legal language inherent in it.

**6.2.2 Enhanced Generalization:**

By incorporating more annotated training examples, the problem of achieving competitive scores across various conditions is encompassed substantially. In the context of the discussed ACD model, generalization is understood as the model’s capacity to extend patterns that have been learned on the training corpus to new, unseen legal clauses. This means there is a decreased tendency of the model to over-fit; that is, to model the noise in the training data and therefore the model performs poorly on new data, Legal text variations are numerous; therefore, a large data set hastens this by providing more samples of text variations typical of real-life legal documents.

**6.2.3 Improved Discriminatory Power:**

Furthermore, the larger and diverse the set of data, the higher the discrimination—which is the model’s capability to differentiate one characteristic or feature from another, in this case between the slight differences and semantic complexities of legal clauses. The differences are observed on the basis of geographical location and norms of legal language as well as the type of clause (for instance, contract provisions, legal liabilities). As a result, the ensemble model can distinguish and sort out these differences properly and thus enhance the stability of clause classification results.

**6.3 Benefits of Diversity:**

The issue of dataset diversity is equally important. Of course, diversity can be also regarded as the presence of legal clauses and their type, registers, and stylistic features. For example, including clauses coming from different legal systems (comparative law – common law vs. civil law) and/or language backgrounds (English, legal language in other languages) enhances the model’s exposure to real language heterogeneity. This exposure becomes the ensemble model’s strength as the model is provided with an opportunity to work on a diverse knowledge level in order to address different legal texts comprehensively and accurately.

**7. Conclusion and Future Work**

This paper presented our automated system of contract comparison, which leveraged NLP and machine learning techniques to facilitate the human review process of legal documents. It offered extremely fair and accurate identification and highlighting of differences that exist in the various versions of a contract, greatly reducing the time needed in the manual review process.

Some future work may then be envisioned, including:

1. Implementation of more advanced NLP semantic understanding
2. Development of domain models trained on legal language
3. Implementation of multi-version tracking for a single contract
4. It assesses the potential of using a named entity recognition system to begin identifying parties and key changes to dates or monetary values.

**8. References**

1. <https://medium.com/@adamhacklander/creating-an-ai-model-to-locate-key-clauses-within-contracts-6b3d7b91cc82>
2. <https://www.kaggle.com/datasets/bahushruth/legalclausedataset?select=duration-of-agreement.csv>
3. <https://www.simplexico.ai/blog/label-legal-clauses-with-a-simple-classification-model>
4. <https://contracts.justia.com/companies/intel-694/>
5. [Contract Management Use Cases for Machine Learning - Analytics Yogi (vitalflux.com)](https://vitalflux.com/contract-management-use-cases-machine-learning/)
6. <https://www.researchgate.net/publication/332171940_Classifying_Semantic_Types_of_Legal_Sentences_Portability_of_Machine_Learning_Models>
7. <https://towardsdatascience.com/how-to-set-up-a-machine-learning-model-for-legal-contract-review-fe3b48b05a0e>