HUMAN-IN-THE-LOOP CLASSIFICATION FOR MULTI-PAGE ADMINISTRATIVE DOCUMENTS

by

Rakshit Makan

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2022

This thesis is dedicated to my family, friends, and Professor Evangelos Milios who supported me in every situation.

Table of Contents

List of Tables

List of Figures

Abstract

Real estate administrative documents present a unique challenge where each condo corporation's documents come from multiple sources and in varying formats. Moreover, each record has varying sizes, making it very time-consuming to read and recognize these documents systematically. A condo buyer must understand these documents before proceeding with the purchase. Our industry partner is a company that gathers and summarizes the condo corporation's documents and highlights essential information that the buyer should consider. In this process, the most laborious task is categorizing these segmented documents where the number of categories is unknown and changes based on the domain expert's knowledge. It is only possible to create an automatic classification system for such challenging data without the involvement of domain experts. So we propose a user-centric, scalable, and reliable multi-page document classification solution for our industry partner using a Human-in-a-loop (HITL) framework. It is a web application with an interactive annotation system for the user to provide feedback at the page and document level to train a two-step classification system. The classification pipeline is a modular system that incrementally learns from the domain expert's page and document-level annotation independently. We evaluated the level of feedback required by the HITL system by comparing our proposed system quantitatively and qualitatively with document-level input via a user study. In the same survey, we also assessed the need for the HITL system in real-world classification problems by comparing the automatic system with our proposed page-level annotation system. Unlike previous research, we explicitly used textual features and state-of-the-art (SoTA) language models to determine the predictive model suitable for real estate administrative documents that can be used in an online setting. In our work, we compare NLP models for multi-class classification for multi-page scanned documents and found that frequency-driven models performed better than deep language models. We demonstrated that the HITL system with page-level annotation ability works better than fully automated classification and document-level annotation for a real-world multi-page document classification problem.

Acknowledgements

I begin by thanking my parents for supporting higher education and believing in my ambition with all their hearts. I would like to all my professors who have taught me during my master's, especially finding innovative ways to deliver incredible lectures online during COVID restrictions. I also want to extend my thanks to the Dalhousie administration for smoothly transferring from online to offline and helping students with unrestricted access to services. Since I arrived in Canada, my friends, Shyam Aghara, Vansh Sharma, and Shivani Gupta, have been family to me. They always inspired me to pursue my thesis and assisted me in coping with personal issues. My colleague at the MALIS lab, Syed Mohammad Baqir Husain, helped me with my thesis from its inception to its final defense. They consistently encouraged and supported me in my efforts. Lastly, I am grateful to my thesis supervisor, Dr. Prof. Evangelos Milios, for his essential counsel, unwavering support, and patience. His vast expertise and extensive experience have inspired me throughout my academic studies and everyday life. Since the beginning of my master's, prof. Milios has always supported my work and experimentation. The timely completion of my thesis was only feasible with his ongoing direction and supervision from its inception through its documentation, mock defenses, and defense. I am appreciative of the time and work he put into this thesis.

Chapter 1

INTRODUCTION

The clients of real estate companies expect their advisors to provide them with all the information related to the condo corporations, including the Bylaws, rules, financial reports, and strata plans. Real estate advising companies gather these documents from each corporation for their clients and create a hand-crafted summary report, which helps them decide whether to buy. There are 15-16 types of different documents accounting for 200 to 300 pages that are scanned and stored in PDFs separately. Currently, the company has a manual process where they OCR documents to highlight the critical text to be added to the summary report. Once they create a pool of digital copies, they manually label these documents, order them to highlight the essential lines, and add a tag. All these steps, as shown in Fig. 1.1, are done for each corporation by a domain expert with deep knowledge of these documents. The whole process is time-consuming and has a lot of labor costs associated with it. There is a necessity for Robotic Process Automation (RPA) which can take these documents as input, recognize their type, and organize them in an order that the company can use to find essential information. The automation process would include an OCR engine, an feature extraction system, a classifier, and some rule-based information extraction processes like table data extraction and date recognition. Since the files are specific to the real estate domain, the output has to be reviewed by the domain expert. Upon the data exploration, we found that the system required for this problem statement has to tackle the following dataset challenges:

1. Scanned administrative text
2. Overlapping information between different types of documents
3. High volume of pages per document
4. Limited and imbalanced data

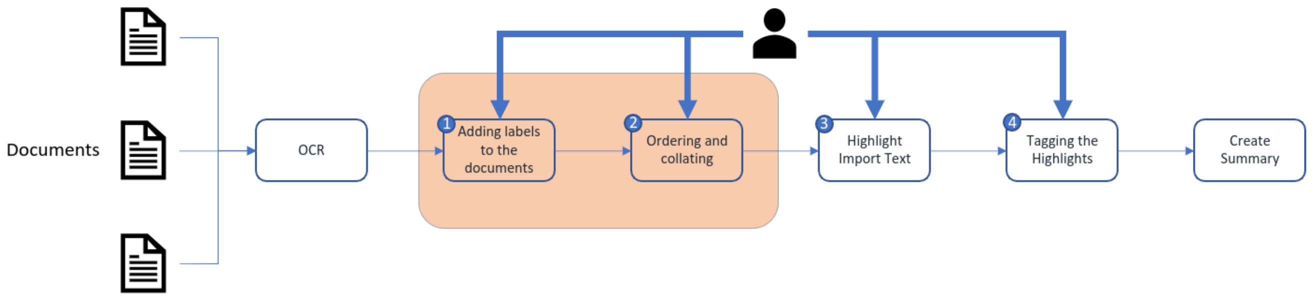


Figure 1.1: **Current Workflow :** The flow chart shows the current steps taken by the company to create the summary report from the document collected from real-estate corporations. Steps 1 to 4 are the manual steps done by the domain expert to formulate the final summarised report. The highlighted step is the most time-consuming for the company, where they have to label the documents and sort them into a particular order, in our case, date-wise.

5. Non-stationary classes

Since no public dataset corresponds to the scanned administrative multi-page real estate documents, which have similar challenges as the company's data, We spend most of our time understanding the data and exploring possible solutions that can adhere to the requirement of real-world business problems. The early research with scanned documents treated each page as an image and used different layout features to create feature vectors [11] [9] [20] [2]. The recent paper used a state-of-the-art multi-modal approach to combining text and visualization [13]. Given the advancement of the natural language model models, the imaging modality shows slight improvement when compared to the uni-modal text models [13]. This motivated us to explore the text-based models for our dataset and cover the following area as part of our research:

1. Which is the best-performing model for the dataset?

2. How can we create a user-centric incremental learning framework?

3. What feedback do we need from the user to improve the model?

Moreover, the dataset used in most of the research for this type of task differs from ours in structure and content, which required us to evaluate our dataset from scratch with classical and state-of-the-art text classification models. Our investigation found that the documents' composition changes drastically from corporation to corporation. For example, the Depreciation Report document may or may not include strata plans and finance documents. Also, the label can change based on the categories defined by the condo corporation and give it a label as MISC for this example. A real-estate expert could identify the accurate label for such complex document structures. This motivated us to create a user-centric classification system by implementing a Human-in-the-loop framework for online training [29]. We used an early approach for the classification model as a baseline where they classified each page and aggregated the result for the document [11]. The use of HITL in NLP for classification tasks is scarce and has been only explored for the clean or short sequence text [16]. Our proposed study explores the following as part of the solution to real-time problems:

1. The need for HITL for scanned administrative documents
2. The level of annotation required when dealing with scanned multi-page documents
3. The qualitative evaluation of the proposed system by real estate domain expert

There are two primary components of our HITL document classification system, similar to any HITL system, an inference pipeline and a training feedback loop, as shown in Fig. [1.2]. The inference pipeline is the one that takes a document as an input and outputs a predicted label for the document. Internally, the document goes through several processes like PDF parsing, data preprocessing, table parsing, data extraction, and a text-classification system. Our text classification is a two-step system with a Page Importance Classifier (PIC) and Main-Page-Classifier (MPC). The main idea behind this classification system is that we only need important pages to classify a document. PIC is a binary classifier that filters out the pages based on their textual information. The MPC only looks at the pages marked as necessary by the PIC to predict a label. Both PIC and MPC get retrained based on the annotations provided by the domain expert at the page and document levels, respectively. Initially, the PIC is a naive system considering everything as necessary, whereas the MPC is trained on the initial dataset provided by the company. A HITL system continuously requires annotated data, and domain experts cannot annotate every data instance. Our proposed solution changes this to a requirement-based system where users provide input on a need basis. The document-level annotation happens during validations, where the user accepts or rejects MPC's output. The page-level

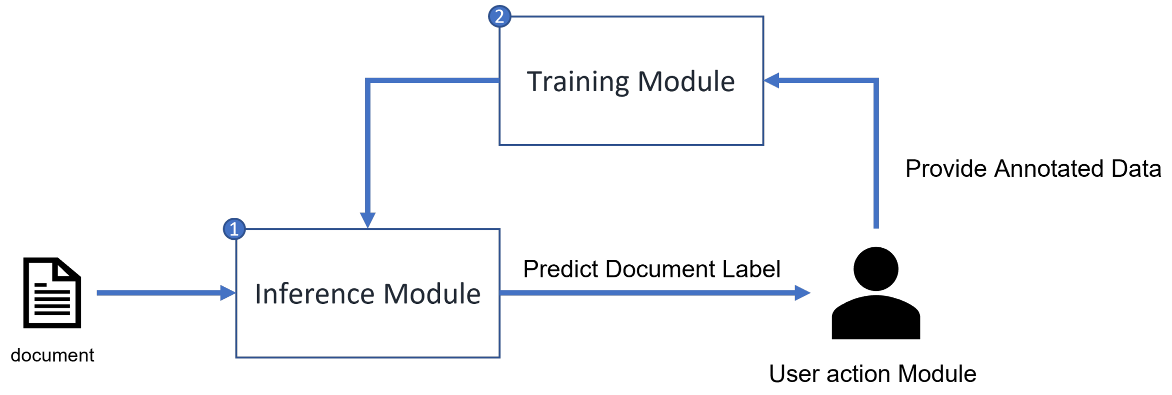


Figure 1.2: **Modules of HITL :** The flow chart points out the challenges in the company's current workflow.

annotation expects the user to mark the page as relevant or irrelevant and update the label of the individual page based on the content. The high-level flow of our proposed methodology is shown in Fig. 1.3. In our paper, we evaluated the system, which has both PIC and MPC, with a system with MPC only. This evaluation also answers the questions about the level of annotation required by the HILT to improve performance. We found that the page-level annotation showed improvement over the document level. The solution has the following advantage over the fully automated multi-page text classification pipeline:

1. **Reliability**: The user and developer perceive the problem differently. The user finds it hard to translate the knowledge and requirements to the developer who is formulating the system. This led to continuous to-and-fro between the teams, leading to unreliable and inferior results. Our solution allows the domain expert to correct the system after a developer finishes the development. Hence, increasing the reliability factor compared to fully automated solutions.

2. **Scalability**: Despite being restricted by domain, the data grows exponentially in real estate, which requires a system that can process as it comes and update the model. Updating a model requires annotated data, data preprocessing, and carefully curated features. We included all three steps in our design to allow the system to scale and adapt as the data grows incrementally.

3. **Interactive Explainability**: Our solution bridges the gap between domain expert knowledge and classification model training. It requires a system that

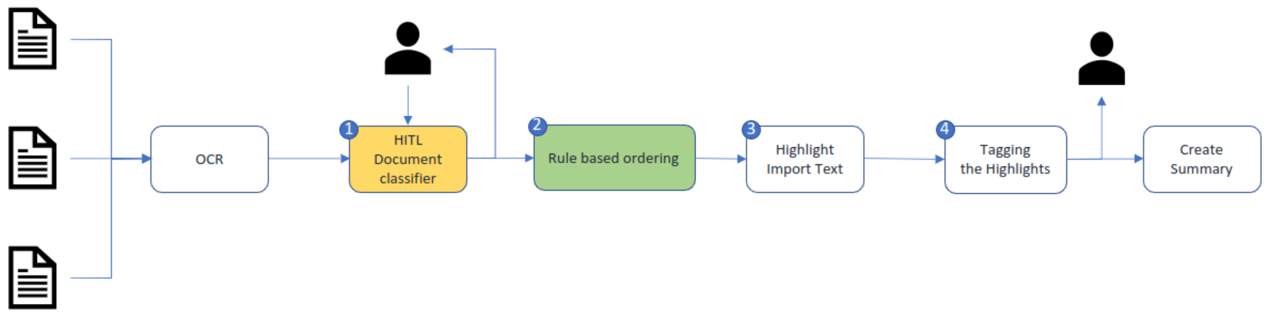


Figure 1.3: **High level Solution :** The flow chart showcases the proposed solution from a high level. In this, we replaced the manual tasks in the current workflow of the company with the human-in-the-loop framework and a rule-based ordering based on the requirements described by the company

monitors the model's performance and shows where it goes wrong. Our solution provides an interactive dashboard where users can not only visualize the results but also correct them and annotate them at multiple levels of the document. It eradicates the need to test the models outside the system.

1.1 CONTRIBUTIONS

The contribution of our system can be summarized as follow:

1. Our proposed system is the first web-based multi-page scanned document classification solution for real estate administrative documents that use the HITL framework and allows domain experts to annotate at both page and document levels.

2. We explored textual features to classify real estate administrative documents using classical and state-of-the-art models. Conducting the comparative study, we found that the classical model with vocabulary-based embedding performs better than pretrained Deep language models.

3. The user study showed that the HITL framework could improve the underlying text classification model for multi-page documents if the annotations are provided at both page and document levels.

Chapter 2

RELATED WORK

This section covers the relevant literature required to understand the various component of our proposed methodology. It includes work from text classification, multi-page document classification, and the use of the HITL framework in natural language processing downstream tasks.

2.1 MULTI-PAGE DOCUMENT CLASSIFICATION

We covered the research in the last decade or so in multi-page document classification. Early studies addressed the categorization of multi-page texts. Every document comprises several pages, such as documents, insurance, and invoices [11]. They suggest a bag of pages technique based on the same premise as the bag of words, in which a document is represented by a histogram that contains the number of occurrences of various types. The approach was tested on two different-sized datasets. The results demonstrate that increasing the number of documents reduced categorization results, with a difference of 21.1% between the first and second databases. Another suggested a document categorization system based on visual layout structural similarity [25]. Column structures and percentages of text and non-text are extracted. This technique does not suggest a mechanism for classifying multi-page texts. It works nicely for single-page documents.

Previous research has shown that textual characteristics surpass visual features in classification and retrieval accuracy. The textual features are not explored explicitly in any of the earlier studies. A combination of TFIDF and SVM in considerable research to perform the classification task to validate the segmented pages from the stream of document [7] [12]. In a multi-page document, the domain focuses more on segmenting the stream of papers and considers classification task secondary [7] [9] [13]. Moreover, all the researcher focuses on either the multi-modal solution by combining textual features with the image features or working purely on image features [9] [13]

[20]. It has been seen that combining visual elements with text either degrades the performance or does not create significant differences in scanned documents [12] [13]. Our work is foundational, where we explicitly work with textual features of multi-page scanned documents and compare all the different types of embedding systems and state-of-the-art transformer models.

2.2 TEXT CLASSIFICATION

In NLP tasks, text classification is a well-researched area because it has a lot of real-world applications. This task has always been at the core of NLP downstream tasks. There have been vast improvements in the past two decades, taking the research from feature-based methods to few-shot transfer learning. The early techniques generally take two steps to perform text classification. The first step is to preprocess the text data and transform it into machine-understandable vectors using techniques such as Bag-Of-Words (BOW) [32], N-gram, Term Frequency-Inverse Document Frequency (TF-IDF) [23], word2vec [18], and Global Vectors for word representation (GloVe)[21]. The second step uses these vectors with generic classification models such as KNN [19], SVM[6], Random Forest (RF) [1], and Naive Bayes (NB) [30]. The initial model's primary problem is that the performance depends on feature extractions. Moreover, these feature-based methods assume that the words in the text are independent of each other. Hence, they cannot capture the target text's structure or context. The introduction of deep learning models revolutionized the field of NLP. Initial attempts with CNN, RNN, and attention helped researchers improve over classical methods. The development of transformer-based models is a milestone in NLP history [28]. The ability to produce contextual embeddings and use pre-trained models for downstream tasks unlocked many applications. The transformer model uses layers of encoders stacked above each to capture the context. These encoders have a self-attention layer that has quadratic time complexity. This limits the number of tokens the model can ingest for a supervised task. Most initial models have a hard limit of 512 tokens which is enough for most tasks, but the longer text might need to capture long-term dependencies [8]. To overcome the short sequence issue, the researchers try to decrease the complexity using variants of self-attention such as sparse self-attention, block attention, window attention, and linearized self-attention. Some popular long sequence transformer models are Longformers[4], Bigbird[31], ETC[3], and Reformer[17].

2.3 HITL IN TEXT CLASSIFICATION

Text classification is a classic problem in NLP to categorize text into different groups. A text classifier is trained on a set of text documents (X) and their categorical labels (Y) to predict the Y value of an unseen X. For example, twitter's sentiment classifier can predict if one review is positive or negative. Many HITL frameworks have been developed for this problem, training a text classifier, recruiting humans to annotate data based on the current model behavior, and eventually retraining the classifier on the larger dataset continuously [29]. Early research provides a HITL paradigm in which users may interactively modify (add or delete) text characteristics and label new documents [10]. In the same research, they extended their methodology by incorporating active learning into their framework by randomly providing data for users to annotate. They intentionally select samples that optimize the projected knowledge gain [10]. Labelers utilizing active learning can annotate fewer data points to achieve the same model improvement as a system using random sampling. The active learning component was further enhanced by including feature (word) sampling in addition to label sampling (documents) [24]. Another research approach created a comprehensive web-based application that supports generic text classification tasks to facilitate the implementation of HITL text classifiers [26]. In addition, researchers have created domain-specific HITL text categorization algorithms. In deployment, for example, a rumor categorization system designed for journalists incorporates model retraining, data collecting, and data annotation [16]. Text classifiers frequently employ the bag-of-words feature. On the other hand, an investigation proposes alternate feature representation to better support a HITL pipeline [15]. We are motivated by the HITL framework used for online learning for domain-specific data. Our implementation of the HITL text classification system takes inspiration from past research, and we suggest improvements by evaluating our method via a user study [27].

Chapter 3

METHODOLOGY

The section breakdown the methodology used for our proposed solution to the company's document classification task. The subsections cover the problem overview, Data Preparation step, Classifier architecture, interface overview, and system design.

3.1 PROBLEM OVERVIEW

The multi-page document classification task is supervised learning, where we want to predict the document's label. Given the challenges of data, our proposed system is a web application based on the HITL framework specially designed to handle administrative documents in real estate. The methodology extends the baseline of the early approach of labeling the documents using page-level vectors [11]. We replaced the visual features used by the model with the textual features and added a novel human-trainable filtering system. The application handles three primary tasks for the machine learning pipeline: 1) data preprocessing, 2) Inferring the document label, and 3) retraining with newly labeled data. The web app is divided into two parts, 1) the backend covers the data storing and parsing algorithm, asynchronous training and retaining, and updating the database with user actions. The front end provides an interface for output validation, page annotation, and an interactive dashboard to make retraining decisions.

3.2 DATA PREPARATION

The data preparation is an integral part of our proposed HITL web application. Our experiment showed that frequency-based embedding like BOW and TFIDF perform well on our real-estate textual data. We created a process to treat the scanned documents, starting from creating page chunks, extracting the text and metadata, and then performing the routine task of text cleaning. The routine step involves removing

	Python Library	**Input**	**Output**	**Documentation Links**
1	TIKA	PDF	Text of the Page	https://github.com/chrismattmann/tika-python
2	PyPDF2	PDF	Remove annotations from the page	https://pypi.org/project/PyPDF2/
3	Tabula-py	PDF	Detect and extract tables	https://pypi.org/project/tabula-py/
4	sutime	Text	Extract time and date from the text	https://github.com/FraBle/python-sutime

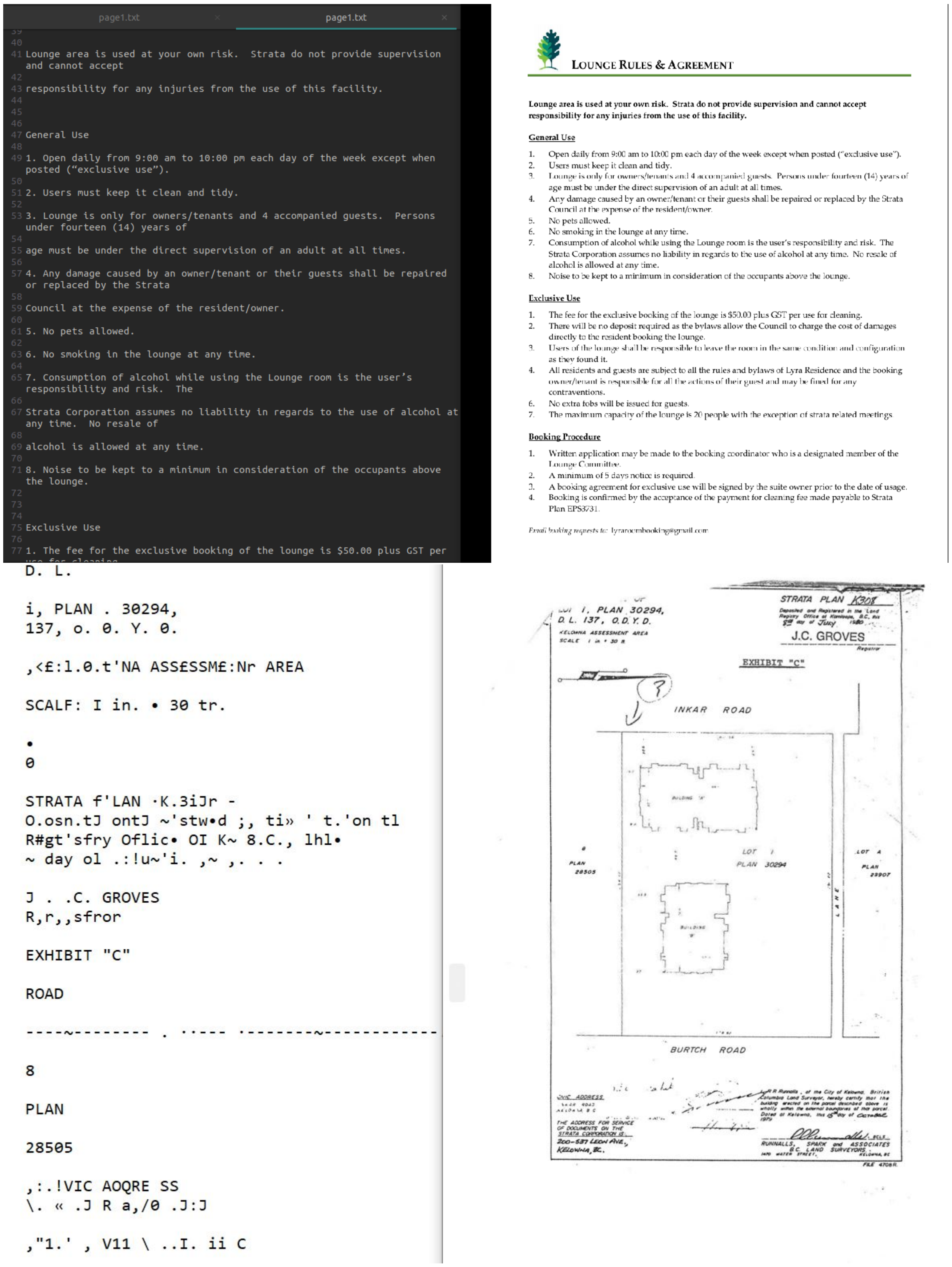
Table 3.1: Python libraries used for extracting the text and other information from the PDF

special characters, numbers, stop words, and extra spaces. Our HITL system repetitively uses a data preprocessing algorithm to prepare data for both inference and retraining. The algorithm uses powerful data extraction and cleaning libraries in the python programming language. We categorize the python libraries concerning their focus of action shown in Table 3.1.

The main idea is to extract the text from each document page. The main challenge is to create a general parsing algorithm for various structures, formats, and fonts. We first started testing various parsing tools available in python. After evaluating multiple outputs, we found that the Apache TIKA [1] library can provide complete textual information on a scanned page. The input and output of some samples can be seen in Fig. 3.1. Some pages only contained tables, and parsing them as text would destroy the context of the information. The tables are significantly populated in documents like Depreciation and financial report. Reading those numbers without the context of the tabular format will be garbage for the system. Strata-plan (SP) is a particular type of document containing many maps and geographical images along the text. We found out that the page containing graphics can add noise to the system, so we removed them using a rule-based filter discussed below. We tried multiple libraries, but for our use case, we found Tabula-py [2] library's table detection and the ability to convert the table into pandas data frame satisfactory. We created a custom algorithm to handle such pages; If the page contains a table, the page will not participate in the classification process; instead, tabular data will be stored in a CSV file. We fixed on following custom rules for the data preparation after experimenting with the process multiple times to enhance the performance.

1. **Number of tokens less than 30**: In our experiments, we found that any text of a page less than 30 tokens does not contain relevant information about

[1]https://github.com/chrismattmann/tika-python
[2]https://pypi.org/project/tabula-py



page1.txt page1.txt

```
Lounge area is used at your own risk.  Strata do not provide supervision
and cannot accept

responsibility for any injuries from the use of this facility.



General Use

1. Open daily from 9:00 am to 10:00 pm each day of the week except when
posted ("exclusive use").

2. Users must keep it clean and tidy.

3. Lounge is only for owners/tenants and 4 accompanied guests.  Persons
under fourteen (14) years of

age must be under the direct supervision of an adult at all times.

4. Any damage caused by an owner/tenant or their guests shall be repaired
or replaced by the Strata

Council at the expense of the resident/owner.

5. No pets allowed.

6. No smoking in the lounge at any time.

7. Consumption of alcohol while using the Lounge room is the user's
responsibility and risk.  The

Strata Corporation assumes no liability in regards to the use of alcohol at
any time.  No resale of

alcohol is allowed at any time.

8. Noise to be kept to a minimum in consideration of the occupants above
the lounge.



Exclusive Use

1. The fee for the exclusive booking of the lounge is $50.00 plus GST per
```

LOUNGE RULES & AGREEMENT

Lounge area is used at your own risk. Strata do not provide supervision and cannot accept responsibility for any injuries from the use of this facility.

General Use

1. Open daily from 9:00 am to 10:00 pm each day of the week except when posted ("exclusive use").
2. Users must keep it clean and tidy.
3. Lounge is only for owners/tenants and 4 accompanied guests. Persons under fourteen (14) years of age must be under the direct supervision of an adult at all times.
4. Any damage caused by an owner/tenant or their guests shall be repaired or replaced by the Strata Council at the expense of the resident/owner.
5. No pets allowed.
6. No smoking in the lounge at any time.
7. Consumption of alcohol while using the Lounge room is the user's responsibility and risk. The Strata Corporation assumes no liability in regards to the use of alcohol at any time. No resale of alcohol is allowed at any time.
8. Noise to be kept to a minimum in consideration of the occupants above the lounge.

Exclusive Use

1. The fee for the exclusive booking of the lounge is $50.00 plus GST per use for cleaning.
2. There will be no deposit required as the bylaws allow the Council to charge the cost of damages directly to the resident booking the lounge.
3. Users of the lounge shall be responsible to leave the room in the same condition and configuration as they found it.
4. All residents and guests are subject to all the rules and bylaws of Lyra Residence and the booking owner/tenant is responsible for all the actions of their guest and may be fined for any contraventions.
6. No extra fobs will be issued for guests.
7. The maximum capacity of the lounge is 20 people with the exception of strata related meetings

Booking Procedure

1. Written application may be made to the booking coordinator who is a designated member of the Lounge Committee.
2. A minimum of 5 days notice is required.
3. A booking agreement for exclusive use will be signed by the suite owner prior to the date of usage.
4. Booking is confirmed by the acceptance of the payment for cleaning fee made payable to Strata Plan EPS3731.

Email booking requests to: lyraroombooking@gmail.com

```
D. L.

i, PLAN . 30294,
137, o. 0. Y. 0.

,<£:l.0.t'NA ASS£SSM£:Nr AREA

SCALF: I in. • 30 tr.

•
0

STRATA f'LAN ·K.3iJr -
O.osn.tJ ontJ ~'stw•d ;, ti» ' t.'on tl
R#gt'sfry Oflic• OI K~ 8.C., lhl•
~ day ol .:!u~'i. ,~ ,. . .

J . .C. GROVES
R,r,,sfror

EXHIBIT "C"

ROAD

----~-------- . ·---- ·-------~-----------

8

PLAN

28505

,:.!VIC AOQRE SS
\. « .J R a,/0 .J:J

,"1.' , V11 \ ..I. ii C
```

Figure 3.1: **Sample Extraction**: The diagram shows the text extraction ability of the Apache TIKA library. There are two types of example in this case. The first one shows the clean extraction of the text, whereas the second shows the noisy extraction where we have special characters and broken sequences.

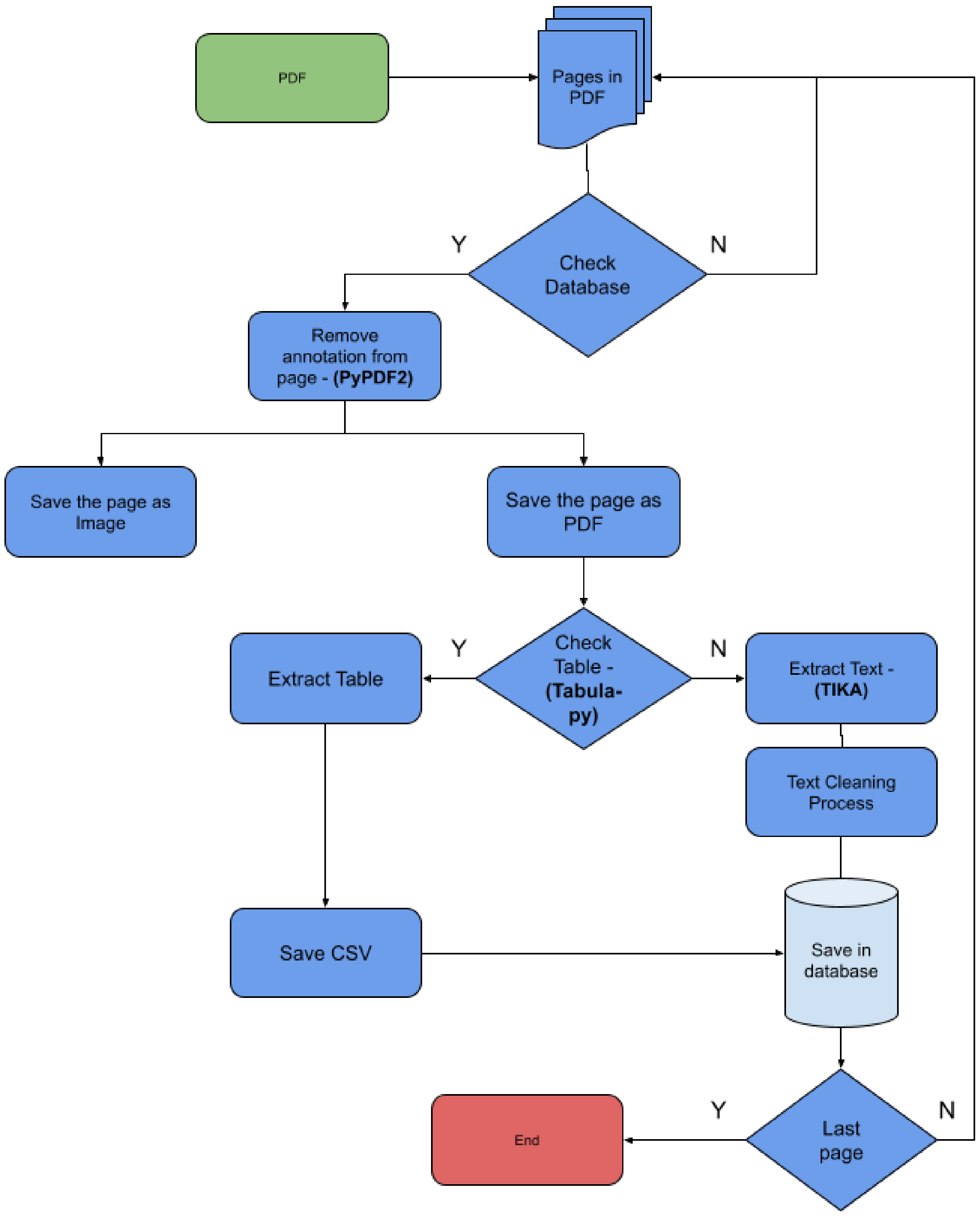


Figure 3.2: **Data Preparation**: The diagram defines the flow of the HITL framework for document and label classification. This user will be controlling the retraining of the model using the "HITL Manager Page" and the associated Model Performance dashboard. The user in the defined framework will validate the recommendation given by the model.

the document or contains graphics. Such pages add unnecessary noise to the data, which can be removed with simple filtering. We apply this filter after the routine preprocessing.

2. **Extracting the Date and time**: The company sorts the document based on the dates if the document belongs to the same category as the condo corporation. In their manual process, the user spends much time finding the date references in the text and highlighting them. We used *sutime* [3] to see them in the text and store them in the database. This data highlights the PDF when downloading the classified PDF using the frontend tool.

3.3 PAGE FEATURE

The text must be converted from its character sequences into a vector space to be used by machine learning models. We consider Sentence-BERT (SBERT) encodings [22] as the feature representation for text documents. SBERT is a variant of the pretrained Bidirectional Encoder Representations from Transformers (BERT) model [8] that offers semantically meaningful encodings for unlabeled text documents. According to studies, SBERT encodings perform better than BERT encoding out of the box in several text classification tasks. Additionally, computing SBERT encodings use fewer resources.

Similar to other BERT variants, SBERT stores a document as an n-dimensional vector of continuous attributes with the following formula

$$x = (a_i, ..., a_n)$$

We use the *bert-base-nli-mean-tokens* pre-trained model, which generates encodings of length n = 768. BERT encodings are limited to 512 tokens, or around 300 to 400 words, because BERT uses subword tokenization. To prevent truncation, we employ the mean SBERT encoding of each unique sentence for our dataset. In a preliminary analysis, we discovered that mean SBERT encodings only benefit the classifiers' F1-Score.

[3]https://github.com/FraBle/python-sutime

3.4 CLASSIFIER ARCHITECTURE

As discussed in Section1, the proposed architecture includes three modules, Training, Inference, and User Action, as shown in Fig. 1.2. These three modules cover the core of the backend of our system and define the frontend aspect. The inference module is a two-step classification algorithm that takes documents as input and provides the predicted label for the document. The user consumes this predicted label to validate and select the relevant pages of the document using an interactive visualization system. The HITL framework dictates that the model will learn the task based on the annotations provided by the domain expert. The training module processes these annotations and retrains the model with newly labeled and historical data to avoid catastrophic forgetting. The main advantage of this system can be summarized as follow:

1. The HITL system allows few-shot learning and incremental learning simultaneously, which means that system can be initiated with limited data and adopt more data and new classes over time.

2. The tool's ability to capture the data changes in real time makes the system robust and reliable over time.

3. It takes away the burden from a Machine learning practitioner to be involved in the process repeatedly. Rather the machine learning algorithm will learn from the wisdom of the domain expert.

4. The tool takes care of imbalanced classes by allowing users to add more instances of smaller classes over time.

3.4.1 TWO-STEP PAGE CLASSIFIER

A multi-page document could contain a long sequence of text. Processing such a long sequence using the state-of-the-art transformer models is impossible because of the restriction of 512 tokens. Even the transformer models claiming to be made for long sequence text [4] [31] can only accommodate sequence lengths up to 4096. The early document vectorizing schemes for creating document-level vectors create high-dimension vectors, which are computationally expensive to train and infer. To

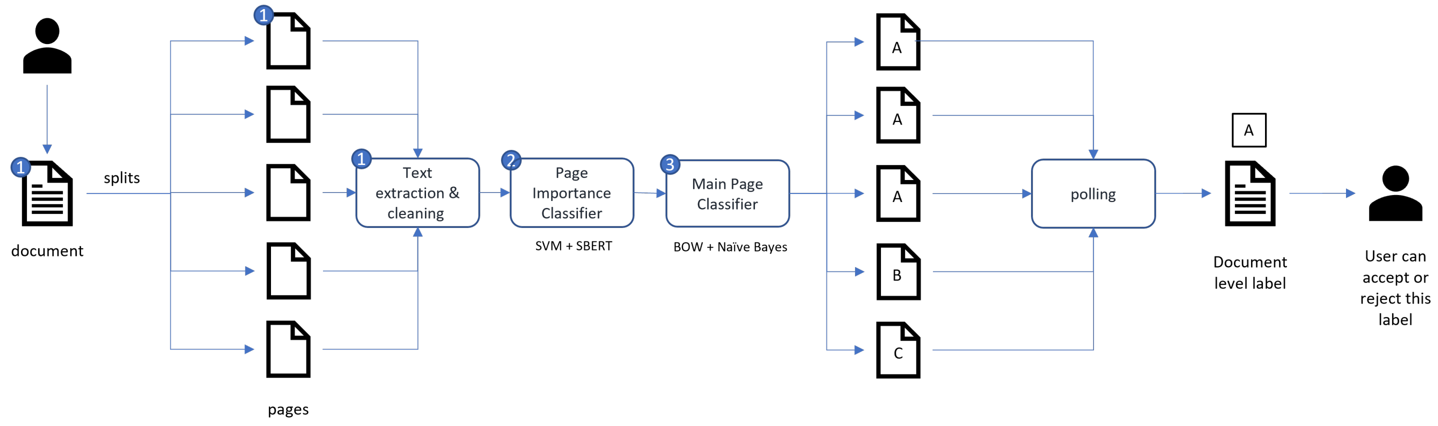


Figure 3.3: **Proposed Pipeline**: The diagram elaborates the pipeline of our proposed system. We expect the scanned PDF from the company will be parsed by our tool and get classified. The ordered documents will go back to the user for the highlighting exercise. The highlighted sentence from the document will be extracted, and a label will be provided. The final steps will remain the same as in the original flow.

improve the performance, we added a few rule-based filters that ignore the pages containing tables and length of sequence less than 30, along with a two-step document classifier that works at the page level that learns to filter pages based on domain expert knowledge. The flow of our proposed document classifier is shown in Fig. 3.3. The document classifier comprises six steps, of which steps 1 and 2 are discussed in section 3.2. Steps 3, 4, and 5 consist of two classifiers, a Page Importance Classifier (PIC) and a Page Classifier, and an aggregation step to produce a final label for the document.

The central idea behind creating the two-step technique is based on a fundamental hypothesis that we do not process all the pages to classify a document. There could be only a handful of relevant pages to the document, which helps the classifier identify the document. However, the principal question is how the system will understand the importance of a page, as machines understand text differently from humans. It will not be possible for a person to mark something important until or unless he has a solid reason. So, a call-to-action system should motivate the user to select document pages that are not useful for the classifier. So, the setup contains two classifiers 1) the Page Importance Classifier (PIC) and 2) the Main-Classifier that allocates labels to the page.

1. **Page Importance Classifier**: This is a binary text classifier acting as a filter

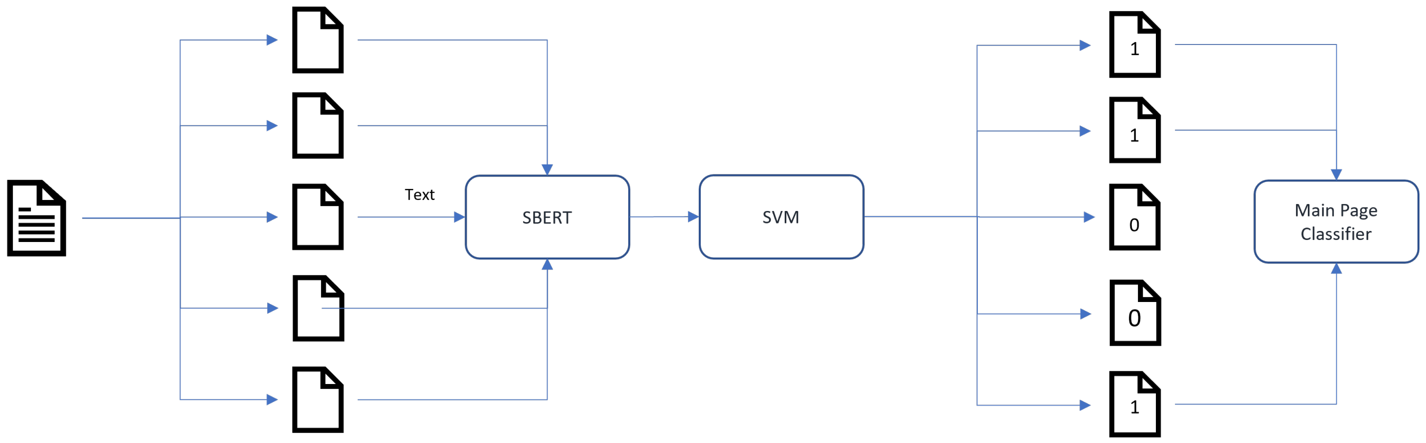


Figure 3.4: **Page Importance Classifier**: The diagram defines the inference flow of the PIC pipeline. The stream of the pages will be labeled as 0 or 1 based on the content of the page. Only the page labeled as 1 will be transferred to the main page classifier or MPC

to the main classifier. Since there was no labeled data to evaluate the page importance classification, we used the SoTA SBERT embedding system with an SVM classifier. As shown in Fig. 3.4, it takes the SBERT [22] embedding and uses the support vector machine with a linear kernel to predict page importance. It restricts the pages the user decides are irrelevant to the main page classifier. When initialized, the PIC understands all the pages as important and allows all pages to go to MPC. The classifier is trained only on the page-level annotations provided by the domain expert using an interactive user interface discussed in Section 3.5. We allow users to preview each page of the document to mark the importance of the page depends, as it allows them to make decision-based on the following three factors:

(a) the quality of the information present on the page

(b) the placement of the page in a document

(c) the contribution of the page in the classification of the document

2. **Main Page Classifier** : It is a multi-class classifier that labels the filtered pages from PIC and uses polling operations to produce document-level labels, as shown in Fig. 3.5. The polling operation sort the labels assigned to the pages by their count and selects the maximum one for the document. This classifier gets trained as the data comes incrementally. To set up the incremental data

pipeline for the classifier, we used the HITL framework. The training data is created based on the user's validation of the model's output and annotation of the documents' pages with overlapping content. The user can visualize the incoming data using a dashboard, as discussed in Section 3.5. Given the use case of real estate multi-page scanned documents, the data is dynamic and highly imbalanced, and the system needs to evolve as the new data arrives. The data for the upcoming training cycle are of two kinds:

(a) **Historical data**: This is the data already known by the classification model.

(b) **New Annotated Data**: The data on which the classifier has made a prediction, and the user has made validation and other annotations.

For every user, the MPC is initiated using the sample data provided by the company, which contains 11 labels. The sample data details are discussed in the upcoming Section 4.1.1. For the historical data, the ground truth comes from the limited data provided by the company, discussed in detail in Section 3.2. We have included a validation step to prepare the training data using the new data set. This step will ask the user to accept or reject the predicted value. Depending upon what the user selects, the data will be recorded into a relational database. Since the SVM is a naive classifier and requires to be retrained from scratch, the pool of new and historical data will be sent to the model for training.

3.4.2 THE HITL FRAMEWORK

The system includes feedback loops that convert user action into training data for our proposed two-step classifier. The are two levels of feedback from the user 1) the document level and 2) the page level based on the prediction provided by the classifier. The system records these two input types and stores them in a relational database at respective levels. Later, the domain expert can train the MPC and PIC from a pool of new training data. There are specific actions the user can perform at each annotation level.

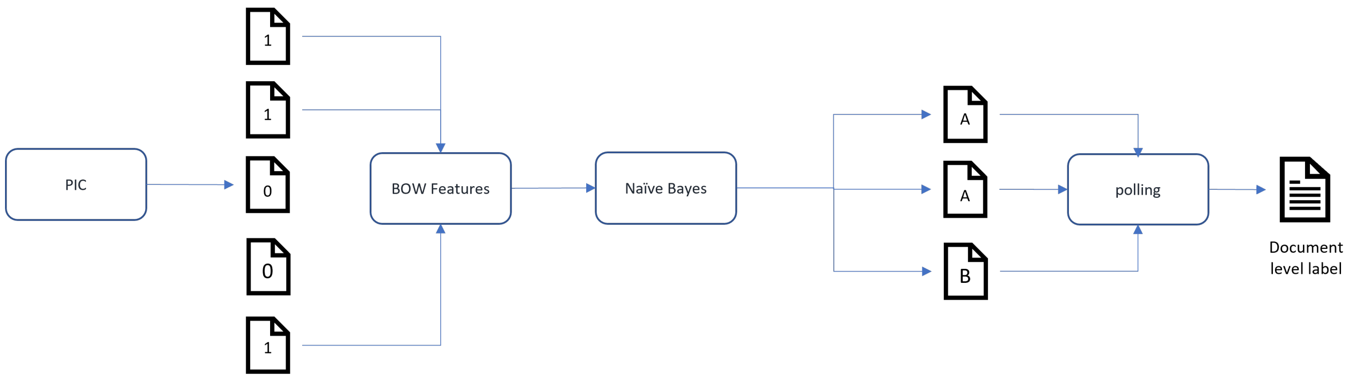


Figure 3.5: **Main Page Classifier**: The diagram defines the inference flow of the MPC pipeline. The stream of the pages will be labeled as one of the classes defined by a domain expert. The polling operation assigns the label to the document.

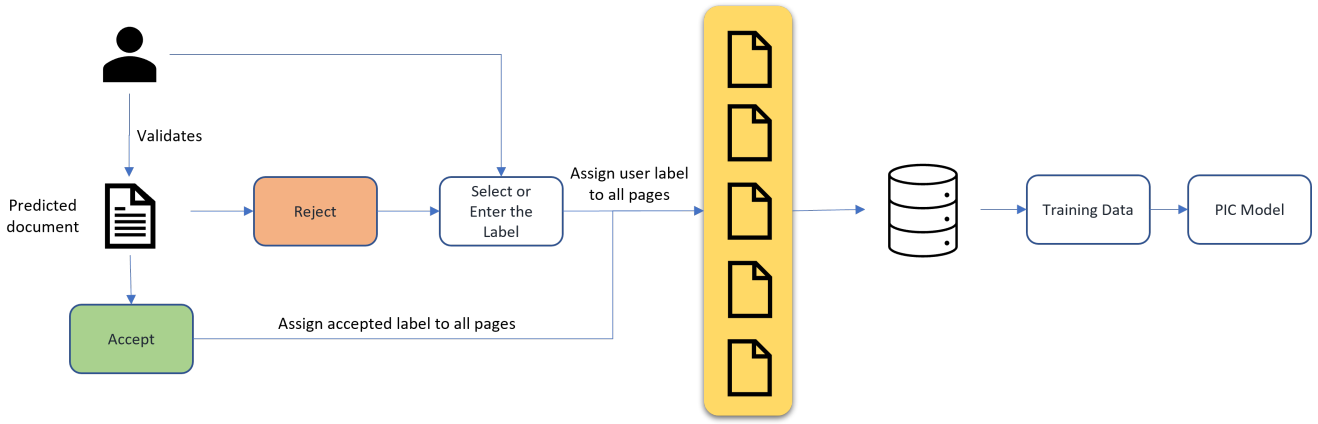

Validates
Predicted document
Reject
Select or Enter the Label
Assign user label to all pages
Accept
Assign accepted label to all pages
Training Data
PIC Model

Figure 3.6: **Document Level Annotation**: The diagram defines the flow of the HITL framework for document and label classification. This user will be controlling the retraining of the model using the "HITL Manager Page" and the associated Model Performance dashboard. The user in the HITL framework will validate the recommendation given by the model.

1. **Document Level Actions:** The document level annotation consists of the validation of the user on the prediction made by the main page classifier 2. During the validation, the user can either accept or reject the output provided by the MPC. The predictions accepted by the user will update the label of each page in the file with the predicted label. On the other hand, the user can reject the output if he perceives the document to be different from the predicted label. Upon selecting the reject option, the user will be given the option to select a label or provide a new label. In either case, the user action will overwrite the predicted label with the document level label, as shown in Fig. 3.6.

2. **Page Level**: The page-level feedback is more granular than the document-level

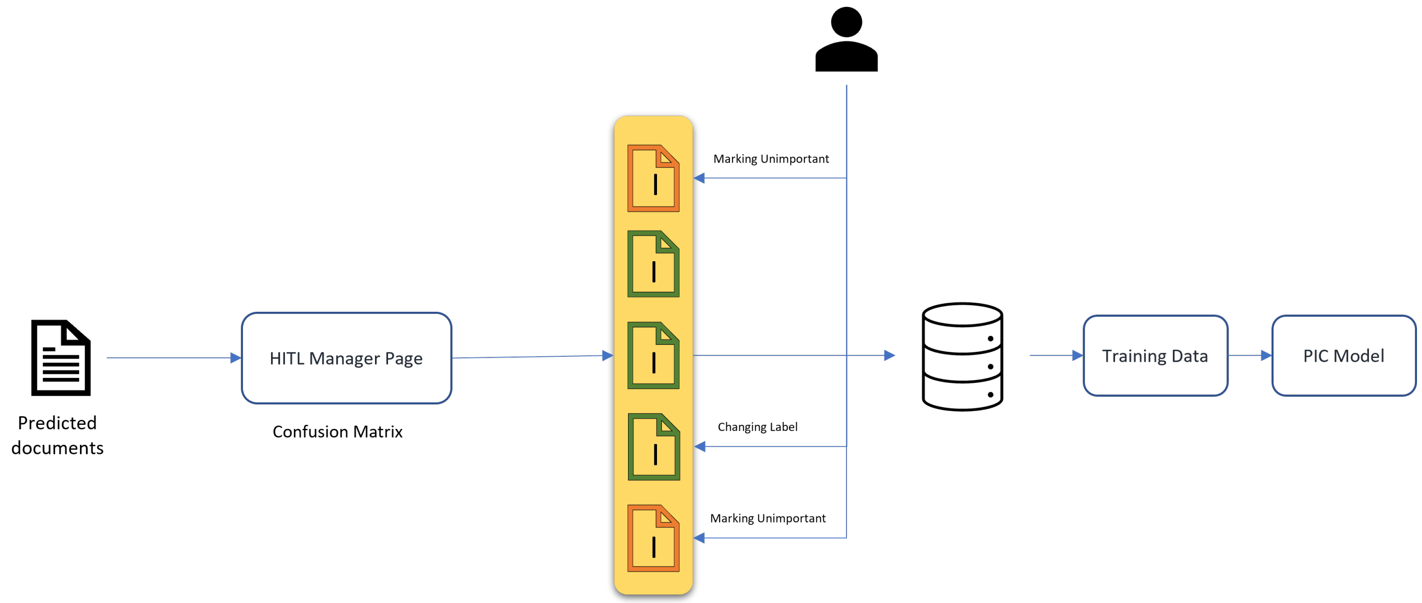


Figure 3.7: **Page Level Annotation:** The flow chart showcases the page-level annotation done by the user. In this, the user can do two types of annotation 1) changing the page's importance in the document and 2) changing the label of pages. All the annotations get recorded into a relational database where page-level data is stored

feedback. The user is asked to select or deselect a page as necessary and fix the content overlap. This task requires a lot of attention from the user and can be tedious. We wanted to make this task as easy as possible so that the user could easily identify data gaps data. As shown in Fig. 3.9, a confusion matrix representation of the classification results was the most acceptable way to list all the irregularities in the predicted data. If the page is correctly classified, it will land on the diagonal of the matrix. It will map the correct label given by the user and the incorrect label provided by the classifier. The user is allowed to explore all the pages in this matrix. To improve this annotation step further, we made the confusion matrix interactable, where the user can check the pages under each matrix tile. Upon clicking, the user can preview all pages with an option to annotate them individually, as shown in Fig. 3.8. There are two types of annotation the user can do at this stage 1) change the importance value from 1 to 0 and 2) change the page's label. The first change will train the PIC model, whereas the second will create data for the Main-Classifier. This feature of page exploration and annotation makes sense because of three reasons

(a) The user will able to directly translate his intuition to the system helping the system to learn the data efficiently.

(b) There could be a possibility where a page belongs to document X, but the information on the page is similar to document Y. In that case, the user can also mark the label as Y instead of X.

(c) It helps in overcoming the data imbalance and later helps in page segmentation tasks where we can know the composition of a document.

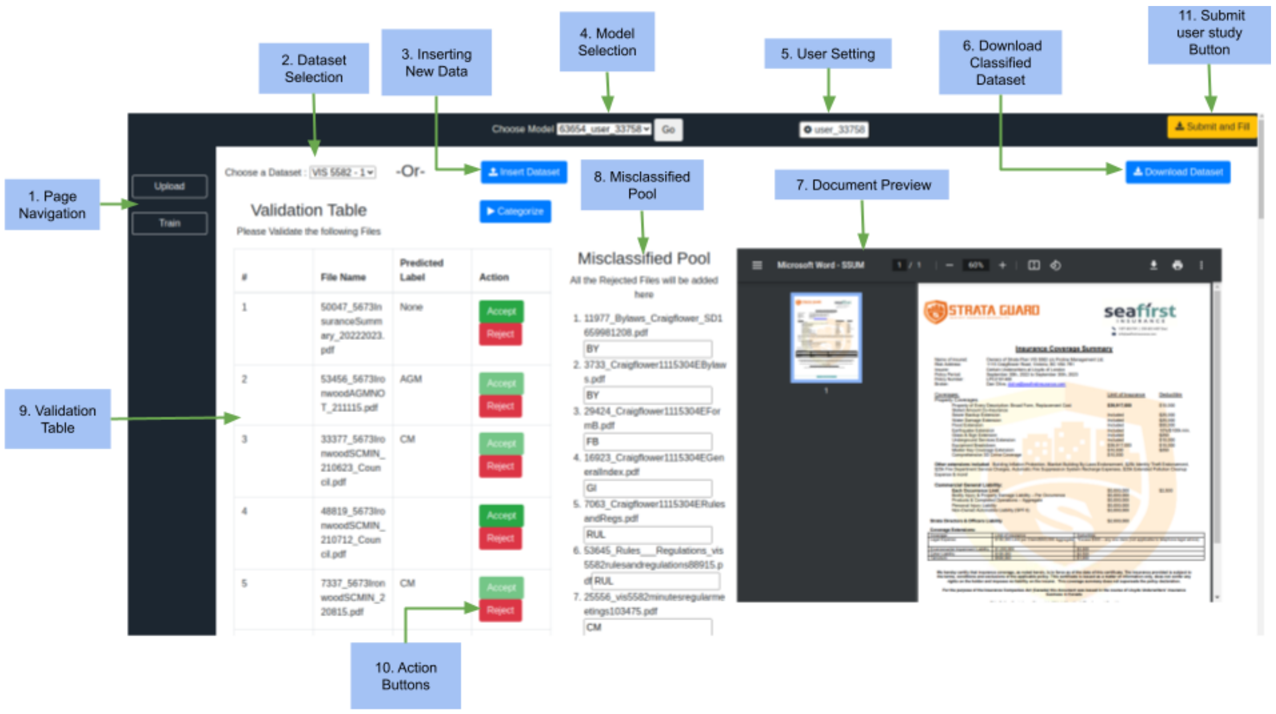
.

3.5 INTERFACE OVERVIEW

The section describes the different components of the user interface and its functionalities. Our system has two main pages 1) the upload page and 2) the Train page. The upload page allows the user to upload the documents, classify them and annotate them at the document level. In contrast, the training page will enable users to have interactive page-level analysis based on the classifier's output and document-level annotations. The component of the user interface shown in Fig. 3.8 is described in Table 3.2 and 3.3.

3.6 SYSTEM DESIGN

This section provides a breakdown of various technologies used to support different components of the framework. Our suggested system is implemented as a web application using back-end technologies like Flask (a micro-service framework written in Python), SQLite (a storage database), and front-end technologies like JavaScript and HTML5.REDIS and session-based user management enable multiple users to work on the system concurrently in a production environment. The design needs secure authentication and session management because the tool is intended to be deployed online.



Validation Table

Please Validate the following Files

#	File Name	Predicted Label	Action
1	50047_5673InsuranceSummary_20222023.pdf	None	Accept Reject
2	53456_5673IronwoodAGMNOT_211115.pdf	AGM	Accept Reject
3	33377_5673IronwoodSCMIN_210623_Council.pdf	CM	Accept Reject
4	48819_5673IronwoodSCMIN_210712_Council.pdf		Accept Reject
5	7337_5673IronwoodSCMIN_220815.pdf	CM	Accept Reject

Misclassified Pool

All the Rejected Files will be added here

1. 11977_Bylaws_Craigflower_SD1659981208.pdf BY
2. 3733_Craigflower1115304EBylaws.pdf BY
3. 29424_Craigflower1115304EFormB.pdf FB
4. 16923_Craigflower1115304EGeneralIndex.pdf GI
5. 7063_Craigflower1115304ERulesandRegs.pdf RUL
6. 53645_Rules___Regulations_vis5582rulesandregulations88915.pdf RUL
7. 25556_vis5582minutesregularmeetings103475.pdf CM

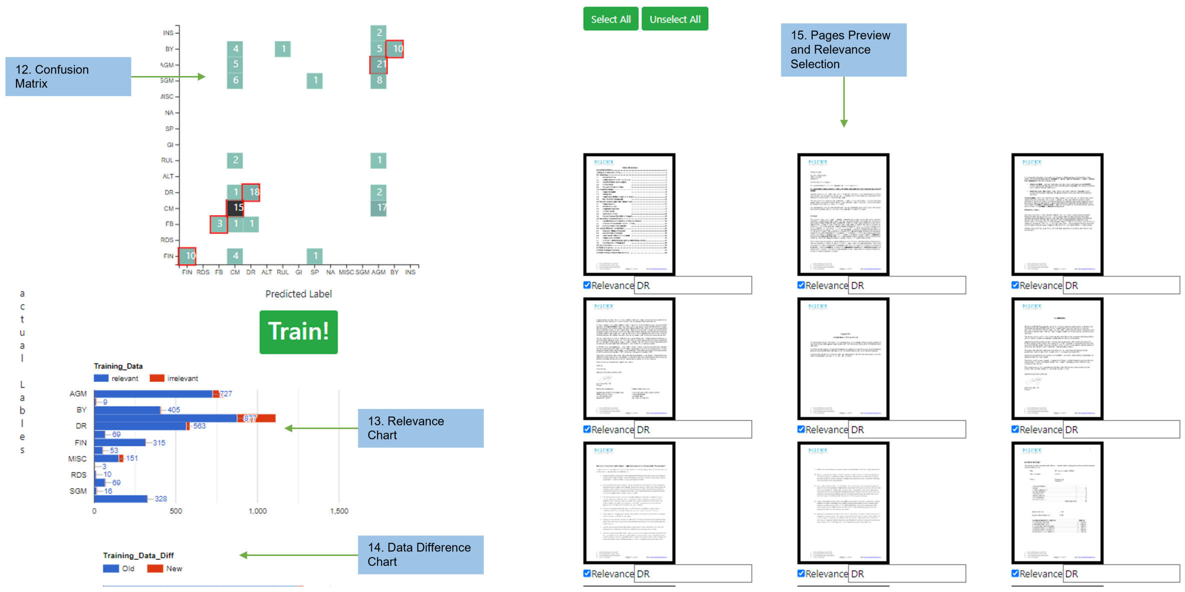


Figure 3.8: **Frontend Screen Shots:** The figure shows the screenshot from the user interface of the flask application used to take feedback from the user. The first image depicts the upload page and document annotation system, whereas the second image corresponds to the training page where the user can train the model and annotate the pages

3.6.1 USER INTERFACE

We rely on our development of contemporary browsers and their rising computational power for visualizations to provide a cross-platform tool that is unrestricted by the user's operating system. Our original CSS and JavaScript codes are the tool's structure and layout managers. We use Bootstrap 5 to create the fundamental user interface components. REST APIs and AJAX are used to connect the front end and back end. All the components in our user interface and their brief description are listed in Table 3.3 and 3.2.

3.6.2 HANDLING THE TIME-TAKING PROCESSES

The front-end system needs to be consistently responsive, so even if the back-end takes a long time to process the data, the user interface should not wait. There are two such processes in our proposed framework (1) classification of the file and (2) training of the model. The time of the task is based on two major factor factors, 1) computational power and 2) the total number of pages to be classified. Within the Flask, we used REDIS to defer these operations by putting information about the task to be performed inside a queue, which gets processed later. We can scale the system horizontally by initiating multiple queues simultaneously. This helped in achieving the multiprocessing for our system and will help handle numerous requests from various users efficiently.

3.6.3 INTERACTIVE CONFUSION MATRIX

The confusion matrix, shown in Fig. 3.9, is obtained using the d3js library in which we created the blocks for each entry in the data. The x and y-axis of the grids represent the actual and predicted labels, respectively. The next step is to place the data on the grid based on the predicted and user-validated labels. We used Scalable Vector Graphics (SVG) to set the height, width, shade, and interactivity. On clicking the block on the matrix, an AJAX POST request queries the SQL database to get the images related to that block. It dynamically shows the pages with the option to mark them as relevant or irrelevant. This confusion matrix explains two things to the user 1) which part of the document does the model unable to understand, and

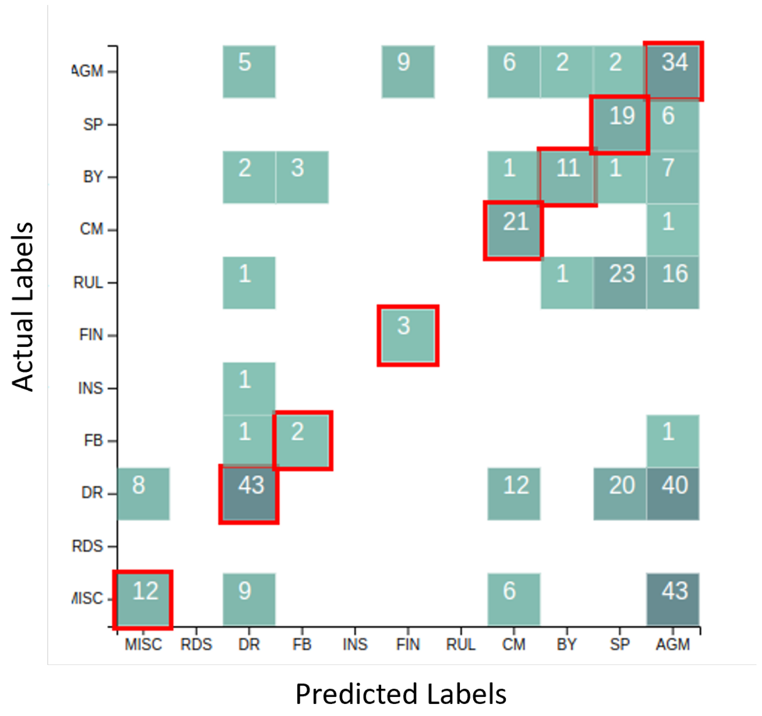


Figure 3.9: **Confusion Matrix**: The diagram is a snip shot of the confusion matrix.

2) which label needs more attention in the next training cycle. This gives the user control over the system and allows him to shape the model in the desired way.

ID	Component	Description
1	Page Navigation	This is the left navigation system where users can switch pages. Our system has two main pages: the upload and training pages.
2	Dataset Selection	Users can switch between the dataset they have uploaded to the system. This helps them to revisit the data they uploaded earlier. Moreover, the dataset is associated with the model it uses to classify.
3	Inserting New Data	The interface allows the user to upload a new dataset into the system. There is no limit on how many files a user can upload into the system.
4	Model Selection	The dropdown allows the user to switch between the trained model. The latest model will be selected by default, which the user can change. This model selection also controls the dataset selection dropdown. The dataset classified by a certain model gets listed under that model name. These model names are auto-generated.
5	User Setting.	This is a drop-down functionality where the user-related setting can be added. As per our tool, we can log out, which clears all of a user's session data.
6	Download Classified Dataset	The user can download the dataset after classifying and validating them. Upon clicking the download button, the back will mark the first found date in the file and highlight it inside the PDF. The downloaded file follows a naming convention in which is \verb+¡file label¿_¡file_id¿_¿¡name of the file¿.pdf+
7	Document Preview	This screen area will display the entire document uploaded into the system. The user must click on the File Name in the validation table.
8	Misclassified Pool	The misclassified pool contains all the files that got misclassified. The user can select the correct label from the dropdown in this section.
9	Validation Table	This table displays all the files uploaded into the system. This table captures the predicted labels and action buttons. The user can preview the pdf, and if the user does comply with the predicted label, they can accept or reject it using the action button
10	Action Button	There are two action buttons 1) Accept and 2) Reject. The accept button will help if the user feels the predicted label is correct else, they click on reject, which will add the file to the misclassified pool.
11	Submit user study Button	The user will click this button after completing the user study to fill the SUS questionnaires and NASA task load index survey

Table 3.2: Components of Upload Page in the user interface based on the marking shown in Fig. 3.8

ID	Component	Description
12	Confusion Matrix	This is a clickable matrix that separates the correctly classified pages from otherwise. This is clickable visualization where the user can preview the pages that are classified correctly or incorrectly
13	Relevance Chart	This is a stacked bar chart showing the number of relevant and irrelevant pages for a class labeled.
14	Data Difference Chart	This is a stacked bar chart that shows the new data added over the old data on which the current model is trained
15	Pages Preview and Relevance Selection	The user can annotate the pages as relevant or irrelevant in this section. They can also change the label of the pages even though their document level label is different

Table 3.3: Components of HITL manager page in the user interface based on the marking shown in Fig. 3.8

Chapter 4

EXPERIMENTS

The experiment aims to evaluate the proposed methodology quantitatively and qualitatively. We started with creating various text classification pipelines considering multiple embedding schemes and simple classifiers. Further, we extended the experiment by fine-tuning state-of-the-art algorithms like BERT and Longformer. The core idea of the first exercise is to study the efficacy of text-based classifiers on scanned administrative real estate documents. Next, we conducted a user study with a company employee to understand the user perception of our proposed system for real-time use. The user study answers multiple questions, such as 1) Does the task of domain-specific document classification require a HITL system or automated pipeline? 2) What level of annotation does a real estate document require to improve the underlying model's performance? and 3) How much impact does the HITL framework have on the classification models quantitatively?

4.1 DATASET

Two types of datasets were used to evaluate our proposed methodology. The first dataset, which contains documents from 15 corporations, was used for the comparative study. In contrast, for the user study, we allowed the user to test the system with unseen live data. Each dataset contains a group of PDF files belonging to a single condo corporation. Since no fixed format is defined for corporations, each group of files has a unique arrangement and a non-static number of labels. Despite the dissimilarities in the structure of the documents, they have some uniform characteristics, such as:

1. All are scanned and contain noise borne from OCR
2. The content overlap among the labels
3. High volume of pages

4. Highly imbalanced

4.1.1 DATASET FOR COMPARATIVE STUDY

To evaluate the models required for the classification of the real estate documents, the company provided documents belonging to 15 different condo corporations. One part of the comparative study is experimenting with document-level and page-level text embeddings with simple classifiers, which require us to formulate two data sets. We started with parsing the text from all the documents and recording them in a CSV file along with their metadata like date, page number, document name, and the document label from its bookmark, provided by the company. Upon compiling the data, we got 14 distinct labels and 2692 pages corresponding to 95 files. Further data analysis revealed that 3 labels, Form V, ID, and budget, have only 1 file each with 82 pages totals, and 40 pages of the extracted pages were blank due to OCR error. We removed these 122 pages, which left us with 2570 rows of data from 92 files and 11 labels, as shown in Table 4.1. As the primary task is to get a document label and compare page-level embedding with document-level embedding, we only calculated the evaluation metric at the document level. We applied data preprocessing steps on the data as described in Section 3.2.

Class Labels	**Number of files**	**Number of pages**
AGM	12	556
BY	10	257
CM	10	459
DR	9	585
FB	9	55
FIN	9	208
INS	7	25
MISC	5	63
RDS	5	11
RUL	7	65
SP	9	286
Grand Total	**92**	**2570**

Table 4.1: Table show the data distribution of the dataset used for experiments

4.1.2 DATASET FOR USER STUDY

For the user study, the domain expert was asked to use three system types: the proposed system with page-level and document-level annotation and two baselines with limited functionalities. We asked the user to integrate the system into their current business workflow and evaluate its performance. To replicate the real-time scenario, the user uploaded and trained different datasets for each system and each iteration. The page count and document count for each system are shown in Table 4.2. The proposed system got the maximum number of documents and pages compared to the baselines. When we look at the label distributions of documents, we can observe that CM and AGM are the highest numbers of documents in all three systems, as shown in Fig. 4.2. On the other hand, the page count of labels shows that AGM, CM, and DR contain a significant number of pages, as shown in Fig. 4.1. During the user study, the user was allowed to add new labels. A new document, ALT, showed up while using the proposed methodology.

Systems	Document Count	Page Count
Baseline 1	111	2902
Baseline 2	109	2899
Proposed System	167	3388

Table 4.2: The table shows the comparison between the page and document count uploaded and trained by the user for three systems.

4.2 COMPARISON OF CLASSIFICATION MODEL

Our paper provides foundational work for classifying multi-page scanned documents for the real-estate domains using the textual feature only. We trained multiple classification models on the company's text data ranging from the early methods like SVM, Random Forest, XGBoost, and Naive Bayes to the SOTA pretrained transformer models. Unlike pretrained models, the early classical model require text to be vectorized before training. There are multiple vectorization schemes for unstructured text data, out of which we used Bag of Words (BOW) [32], TFIDF [23], Doc2vec [18], and a recent SoTA unsupervised vectorization scheme SBERT [22]. SVM + TFIDF have been used previously in multiple research. So we started with the method considering

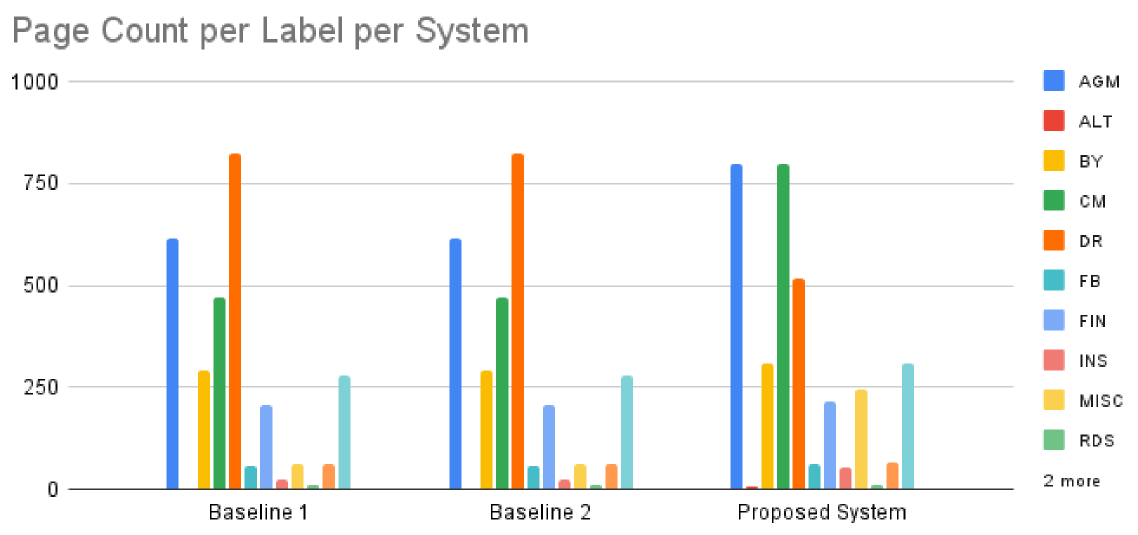


Figure 4.1: **Page Count for each System**: The bar chart compares the page distribution for each label in all systems.

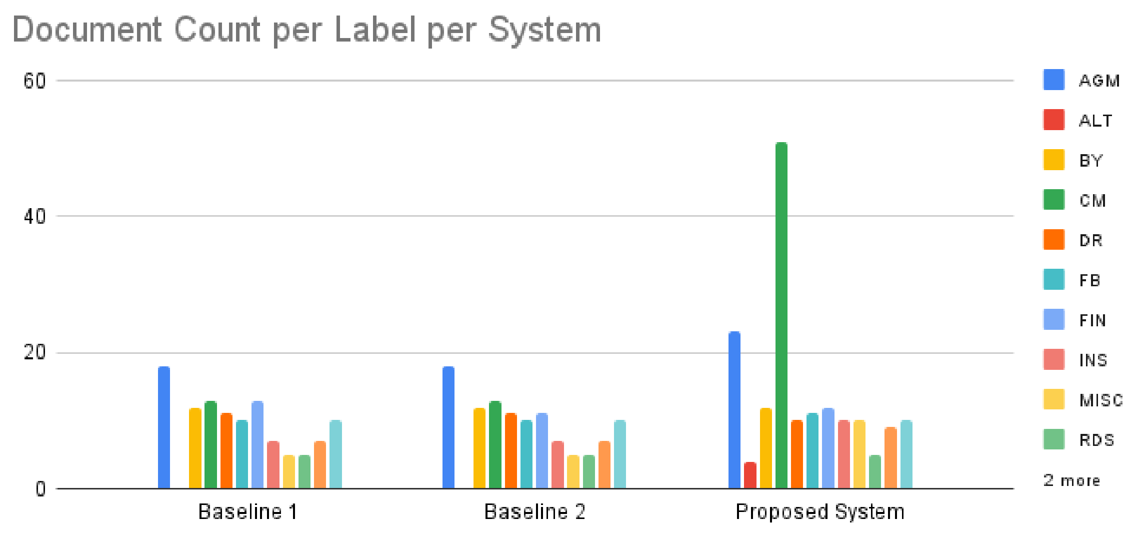


Figure 4.2: **Document Count for each System**: The bar chart compares the document distribution for each label in all systems.

it the baseline [7] [12].

The first goal of the experiment is to evaluate the page-level embedding against the document-level embedding. The page-level data contain 2570 pages with 11 labels that can be mapped to 92 documents. We used all four types of embeddings and trained the simple classifiers. We employed stratified k-fold cross-validation, where k=10, to train the classifier with default settings provided by the sci-kit learn python library. We added a custom algorithm to derive document-level results from page-level results by polling, as shown in Fig. 3.5. On the other hand, the document-level embedding system does not need the aggregation step involved in the page-level system. Another difference from the page-level pipeline is that we did not use SBERT due to its limit of 512 tokens; for the document level, all the instances were longer than that. To compare the performance of the models with the finite imbalanced data, we used weighted F1-score and the average training time. We used the stratified K-fold-cross validation, where k=10, technique with classical models to get the learning curves and calculated the standard deviation.

The second goal is to compare the performance of the pretrained deep language model (DLM) with the best of the simple classifier. We choose two categories of transformer models, short sequences such as BERT [8] will full attention and long-sequence such as long-former [4] with sparse attention. Since the K-fold-cross-validation doesn't work with the deep language model, we used stratified test and train split to divide the dataset into 80-20 ratio and ran the transformer model for 20 epochs.

The data prepossessing was done on the personal system of the lead researcher with i7-7500U @ 2.70Ghz, 16 GB RAM, and 4 cores, which are optimized for CPU usage. The model training involved transformer training and computing more significant vectors like BOW and TFIFD for a document. We used compute Canada's multicore GPU clusters to run our experiments. The PyTorch [1] and transformer's [2] library were used to fine-tune the transformer models using the trainer class that allows multi-processing among the GPUs. The fine tuning step involve training of pretrained model on the our domain specific data, which will update the weights of all the layer accordingly. We used the pretrained models from the hugging face library. In fine-tuning, we let all the layers of the transformer participate in the training process. We

[1]https://pytorch.org/

[2]https://huggingface.co/docs/transformers/index

optimized the models using the Adam optimizer with a learning rate of 0.005 and a batch size of 8. For simple classifiers, we used the scikit-learn [3] library with their default parameters. The details of each model are given in Appendix A.

The outcomes of this task can be summarized as follow:

1. Evaluating the page-level and document-level representation with simple various simple classifiers using weighted f1-score and standard deviation

2. Evaluating the SoTA transformer models with the best simple classifier for real estate multi-page scanned document classification task.

4.3 USER STUDY

We conducted the user study to understand the need and usability of our proposed HITL system for the real estate scanned administrative multi-page documents. The system's usability is evaluated quantitatively and qualitatively in the user study. This study will answer the following research questions:

1. What is the effectiveness of the HITL system in solving real-world business problems compared to fully automated machine learning pipelines?

2. What is the level of annotation (Page or Document) that the user requires to create a sustainable and reliable model for its domain data?

3. Does online training by the domain expert improve the performance of the underlying model?

4.3.1 SETUP

Our proposed system is the first web application to implement the HITL system for multi-page document classification for the real estate domain. As per our knowledge, no other tool can be a potential baseline, leading us to create our baseline tools. Since our methodology promotes the role of a domain expert in the machine learning pipeline, the logical baseline should be an automated document classification pipeline.

[3]https://scikit-learn.org/stable/

Functionalities	**Proposed System**	**Document Level Annotation**	**Automatic System**
Page Annotation	Yes	No	No
Output Validation	Yes	Yes	No
Interactive Confusion Matrix	Yes	No	No
Helper Visualizations	Yes	Yes	No

Table 4.3: The table shows the comparison between the three systems we used for the user study based on the functionalities.

The baseline system only infers the labels for each page and provides aggregated document-level prediction to the user with any insight.

The other aspect of the tool is to compare our proposed page-level annotation with the document level, which is the baseline in this case. The document-level annotation tool should only allow the user to validate the document label and retrain the model with newly added data. In contrast, our proposed page-level annotation allows the user to mark the page's importance and change the label. Using this approach, we got 3 systems, as shown in Table 4.3, for the user study based on the research question we wanted to answer. All the tool versions use the combination naive Bayes + BOW as the main page classifier. The naive Bayes was chosen based on the comparative study performed before the user study. Since there was no labeled data to evaluate the page importance classification, we used the SoTA SBERT embedding system with an SVM classifier.

The user study was performed with a single user from the company who was familiar with documents belonging to the real estate domain. The user was unfamiliar with the tool, which helped us tackle the bias. We asked the user to work with the actual dataset that the company will use in day-to-day work instead of the dummy data and expected him to run the five iterations with a different data set. There was no cap on the number of files uploaded to the tool for classification. Each iteration consists of 44 multi-page documents comprising 300-400 hundred pages. As a metric for evaluating usability, we employed the System Usability Scale (SUS) metric [5] and NASA task load Index [14]. Apart from the quantitative metric, we asked the user to comment on the tool's usability advantages, shortcomings, and future improvements they expect.

Chapter 5

EXPERIMENTAL RESULTS AND DISCUSSION

The domain-specific multi-page scanned documents behave differently from the regular synthetic dataset used in research papers regarding noise level, vocabulary size, and text length. It requires a robust and reliable solution to handle these real-world data challenges and adapt to the documents' rapid changes. The SUS questionnaire results, calculated in Table 5.1, show that the human-in-the-loop system is more suitable than a baseline automatic system. There is an improvement of 20% in the usability score for our proposed method over the baseline. The user found the baseline version of the tool highly unusable for daily use when explicitly asked about the frequency of use.

The system reliability factor was evaluated by a question *"I felt very confident using the system"* in the SUS survey. The proposed methodology was rated the highest among the two baselines. The two baselines were given the same score, which shows that the user feels more confident training models with page-level annotation than document-level. For a non-technical domain expert, a system with more granular information is better than a system showing results at a high level and having limited customization options. At the same time, providing feedback to the system can be mentally demanding and overwhelming for a new user. In the NASA TLX survey, shown in Table 5.2, the user feels more frustrated when asked to provide annotation at both page and document levels compared to its baselines.

The system's performance is directly related to the accuracy of the model used for the prediction. Our experiment shows that the simple classification model, such as Naive Bayes, performs better than state-of-the-art fine-tuned models. The main reason is that the transformer models are not good with limited and noisy data, whereas the feature creation step for the simpler models reduces noise significantly. The same weakness of the deep language model is exposed when we compared the BOW with the SBERT vector scheme with various estimators. The results, shown in Table 5.3,

SUS Questions	**Proposed System**	**Baseline 1**	**Baseline 2**
I think that I would like to use this system frequently	1	1	0
I found the system unnecessarily complex	4	1	4
I thought the system was easy to use	3	3	2
I think that I would need the support of a technical person to be able to use this system	3	4	3
I found the various functions in this system were well integrated	2	1	1
I thought there was too much inconsistency in this system	2	1	1
I would imagine that most people would learn to use this system very quickly	2	3	2
I found the system very cumbersome to use	2	2	3
I felt very confident using the system	2	1	1
I needed to learn a lot of things before I could get going with this system	3	3	3
Final Score	**60**	**50**	**50**

Table 5.1: The table shows the score for each SUS question asked to the user after they used each version of the tool. Baseline 1 is the document-level annotation system created to compare the annotation levels, whereas baseline 2 is the automated system to evaluate the need for the HITL system. The points given for each question asked to the user range from 0 to 5, where 0 strongly disagrees, and 5 strongly agree.

Questions	**Interpretation**	**Proposed System**	**Baseline 1**	**Baseline 2**
How mentally demanding was the task?	Mental Demand	25	25	**15**
How physically demanding was the task?	Physical Demand	5	5	5
How hurried or rushed was the pace of the task?	Temporal Demand	25	25	25
How successful were you in accomplishing what you were asked to do?	Performance	75	**85**	**85**
How hard did you have to work to accomplish your level of performance?	Effort	45	45	45
How insecure, discouraged, irritated, stressed, and annoyed were you?	Frustration	25	25	15

Table 5.2: The table shows the score for each NASA Task Load Index question asked to the user after they used each version of the tool. Baseline 1 is the document-level annotation system created to compare the annotation levels, whereas baseline 2 is the automated system to evaluate the need for the HITL system.

Models	**Vector Scheme**	**F1 - PLA.**	**Std. PLA**	**F1 - DLA**	**Std. DLA**
Naive Bayes	BOW	**98.42%**	0.77%	48.65%	13.10%
	Doc2Vec	57.13%	6.77%	67.64%	10.97%
	SBERT	65.51%	5.46%	N/A	N/A
	TFIDF	98.33%	**0.75**%	43.66%	14.28%
Random Forest	BOW	95.60%	1.88%	79.78%	13.60%
	Doc2Vec	67.38%	4.18%	70.39%	9.67%
	SBERT	71.19%	6.24%	82.77%	11.32%
	TFIDF	96.03%	1.71%	N/A	N/A
SVM	BOW	95.31%	1.48%	77.30%	13.95%
	Doc2Vec	71.62%	3.55%	82.29%	14.01%
	SBERT	78.54%	3.36%	85.72%	8.77%
	TFIDF	94.78%	1.39%	N/A	N/A
XGBoost	BOW	93.13%	1.49%	81.15%	10.47%
	Doc2Vec	69.66%	5.54%	69.43%	13.05%
	SBERT	75.10%	4.13%	N/A	N/A
	TFIDF	17.82%	5.55%	79.15%	16.19%

Table 5.3: Performance of the classical models reported on the proprietary dataset shared by the company when using page-level vectors. The table shows the weighted f1-score at both the document and page levels. We used acronyms in the table to define the system where PLA stands for Page-Level Annotation, and DLA stands for document-Level Annotation. std. denotes the standard deviation in f1 scores for the systems.

show that the SBERT, which was trained using cosine similarity, has consistently underperformed compared to the classical methods like BOW and TFIDF. The SBERT is pretrained to capture the similarity between the two sentences using the cosine similarity loss. This strategy works against the SBERT embeddings where documents have information lap and have similar sentence structure. The BOW and TFIDF are vocabulary-based methods, and domain-specific documents tend to have a limited vocabulary. This could be another reason for a more straightforward approach like naive Bayes to perform exceptionally well over pretrained language models, as shown in Table 5.4.

The company dataset belongs to a restricted domain of real estate text, where the text semantics is uniform across the labels. The information overlap within the documents creates challenges for the classifier to separate the document. The T-SNE plots of document level representation, shown in Fig. 5.1, fail to map similar documents in the same clusters. The page-level representation improves over the

Model	weighted F1- Score
Best Baseline (Naïve Bayes - BOW)	**98.42%**
Longformer	88.60%
BERT	88.91%

Table 5.4: Performance of the Transformer models compared with the best of classical models reported on the proprietary dataset shared by the company using page-chunking of documents. The table shows the weighted F1-score at the document level

document-level representation, but the cluster distributions are highly cohesive to the high information overlap among the documents. The same translates to the classification model's F1 score at both levels presented in Table 5.3. The PIC could solve this to some extent, as Fig. 5.2 shows the quantitative analysis of the user study, the dip in the performance of the automatic ML, and the Document annotation version. The Proposed model gained accuracy as the user provided annotation at the page level.

The same pattern is observed in the learning curve of each label uploaded during the user study showing an overall positive change in accuracy for all the uploaded documents for the proposed system. The user during user-study tested different datasets, as discussed in Section 4.1.2, for all three versions of the tool where the dataset could be similar or completely different from the dataset used to train the underlying classification model. Despite this challenge, the HITL system with page-level annotation handled the changes and improved results over the base variants proving its robustness and reliability.

5.1 DISCUSSION

In this section, we elaborate on the results and discuss the rationale behind our outcomes. We have divided the section into two parts. The first talks about how we overcame the real-world data challenges, and the second talks about the model used and the HITL framework requirement.

5.1.1 Dataset Challenges

The dataset from the company came with a lot of challenges, as discussed in Chapter 1. The purpose methodology's components handle these challenges, like the data

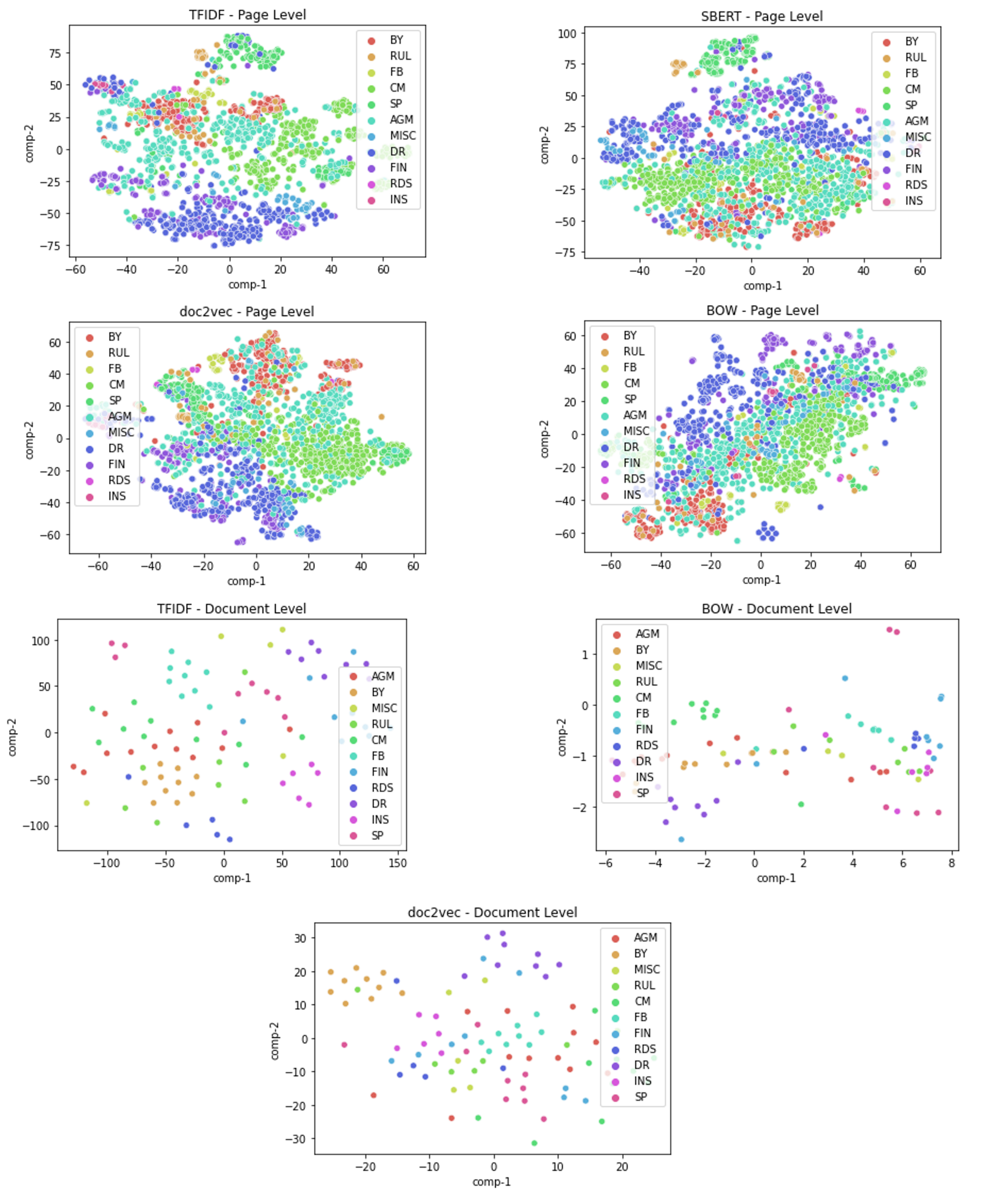


Figure 5.1: **t-SNE Projection :** t-SNE plot of projections of page and document embeddings. Starting from top left to right, page-level embeddings using TFIDF, SBERT, DOC2VEC, and BOW, followed by document-level embedding using TFIDF, BOW, and DOC2VEC

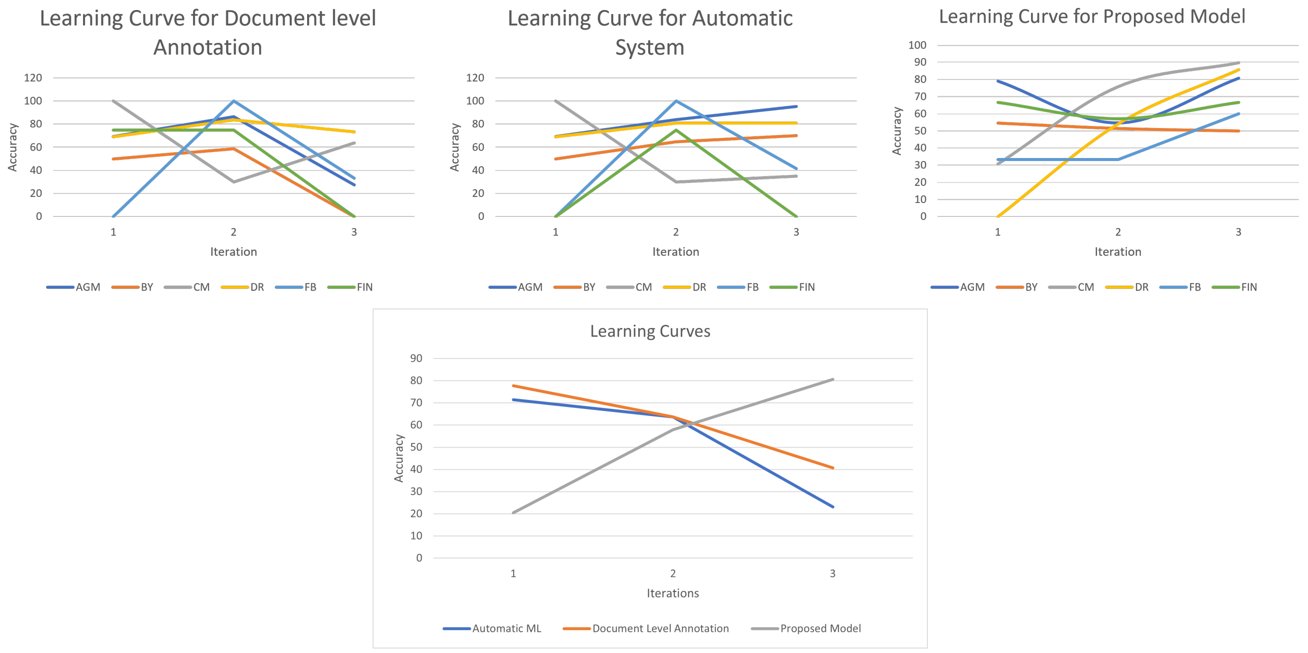


Figure 5.2: **Learning curves** Quantitatively comparing user study results from the accuracy point of view. The graph shows the accuracy attained after each iteration. The First 3 line chart shows the label-wise accuracy trend between the document-level annotation, fully automatic, and proposed model. The single chart in the second line represents the overall learning curve of three systems

preprocessing engine, annotation, and training module. HITL's core strength is that it allows users to provide input directly to the model. The primary challenge was that there was minimal knowledge about the domain. So HITL system allowed us to give control to the domain expert limiting the effort of deep diving into the domain. Another related challenge was that the domain expert needed a system where he could add new classes based on the document's content. The ability to add a new label to the system using the document-level annotation and training module solves this problem. Moreover, the user can keep track of the classes and the number of pages associated with them on the training page shown in Fig. 3.8.

Another two challenges are related to each other. The information overlap and unbalanced classes are co-dependent. We wanted a system that could correctly identify the class of pages in our system so that classes with a low number of pages get more training data. To solve the imbalance class problem, the information overlap should be solved, which means that even though the document belongs to class A, the page belonging to X, Y, and Z classes should be correctly identified. The page-level annotation module handles this issue by allowing users to change the labels of the pages and marking the pages unimportant. By allocating the correct class to the pages, there will be a shift of instances from higher to lower class, solving the imbalance-class problem.

5.1.2 HITL framework

The human-in-the-loop framework's simple idea is to place human beings in machine learning to improve the predictive model. Given the history of the HITL in NLP, the following are the requirements for the online HITL system:

1. There should be an aspect of the pipeline completely or partially dependent on the user.
2. The model should be able to utilize the annotations or user input to update the model.

In our system, we satisfy both the requirements of the online HITL system. We incorporated the domain expert to validate the model's output and engineer document pages based on their importance. Secondly, we capture all the inputs the user provides into a database and later convert them into valuable training data.

Chapter 6

CONCLUSION

The research presents a human-in-the-loop classification system for scanned administrative multi-page documents related to real estate. We began by exploring the company's dataset to understand the nature of the documents and the features required to be extracted and create a robust text extraction plan discussed in Section 3.2. The structure of administrative documents in real estate is highly dynamic and depends majorly on the way condo corporations curate them. The document's content can be completely different regardless of the same class. The only way to define a class for a document is by looking for relevant pages. The relevant pages can be anywhere in the file, and only a domain expert can identify those pages. Therefore, the role of a domain expert is crucial when dealing with real estate administrative documents or something similar.

The domain-specific data behaves differently from the general text used in multiple types of research related to the downstream tasks in NLP. We have seen state-of-the-art results using the transfer learning approach. Still, our experiments show that the classical model with frequency-driven embedding works well for limited domain-specific data. Our experiments show a difference of almost 10% between the SOTA deep language model and the best-performing classical model, as shown in table 5.4. This vast difference indicates that the text data with limited vocabulary and overlapping context requires a classical model that accounts for the composition of the text over the context. The context of the text plays an essential part in training the transformer models, which was missing from the text. To improve the performance of the transformer model, a methodology to capture long-range context is required.

When using a user-centric framework, such as HITL, it is essential to make the model output explainable with the proper interaction and visualization. The primary need for a HITL system is getting users' feedback to improve the underlying model.

Creating an engaging system where users can annotate without feeling an extra burden becomes essential. In our case, we tried two levels of annotations, page-level, and document-level, for our user study involving the real estate domain expert. The document-level annotation is straightforward, where the user has to provide a correct label for the misclassified documents. The page-level annotation is more exerting for the user as it asks to select the page's relevance and change its labels. Comparing the usability score, such as NASA TLX and SUS, with quantitative trends (Fig. 5.2) between the two systems shows a clear trade-off between UX and performance. A non-technical business user prefers an easy system with the moderate performance to a complex system with high performance. Hence, for the user-centric application, the UX components such as placement, the number of clicks, and the direction to use plays significant role in usability.

The software usability survey evaluates a tool based on multiple factors and provides insight from the user's perspective. If we look at the overall score of SUS in Table 5.1, the proposed methodology has an upper edge compared to the baselines. But if individual factors are evaluated, the proposed method shows weak performance in critical areas. The user strongly agrees that the system is unnecessarily complex compared to baseline 1, where the user annotates the misclassified document. This could be possible because the number of annotations increased rapidly when using a page-level system. The only areas where the tool shows an edge over the baselines are ease of use, system integration, and the confidence provided by the tool to the user. The granularity of the annotation and complete control over the training data is the primary reason for these results.

Bibliography

[1] Proceedings of 3rd international conference on document analysis and recognition. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages iii–, 1995.

[2] Onur Agin, Cagdas Ulas, Mehmet Ahat, and Can Bekar. An approach to the segmentation of multi-page document flow using binary classification. volume 9443, 03 2015.

[3] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers, 2020.

[4] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]*, December 2020. arXiv: 2004.05150.

[5] John Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] Hani Daher and Belaïd Abdel. Document Flow Segmentation for Business Applications. In *Document Recognition and Retrieval XXI*, pages 9201–15, San Francisco, France, February 2014.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[9] Ignazio Gallo, Lucia Noce, Alessandro Zamberletti, and Alessandro Calefati. Deep Neural Networks for Page Stream Segmentation and Classification. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7, November 2016. 3 citations (Crossref) [2022-08-21].

[10] Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. Document classification through interactive supervision of document and term labels. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD '04, page 185–196, Berlin, Heidelberg, 2004. Springer-Verlag.

[11] Albert Gordo and Florent Perronnin. A Bag-of-Pages Approach to Unordered Multi-page Document Classification. In *2010 20th International Conference on Pattern Recognition*, pages 1920–1923, August 2010. 10 citations (Crossref) [2022-08-21] ISSN: 1051-4651.

[12] Albert Gordo, Marçal Rusiñol, Dimosthenis Karatzas, and Andrew D. Bagdanov. Document classification and page stream segmentation for digital mailroom applications. In *2013 12th International Conference on Document Analysis and Recognition*, pages 621–625, Aug 2013.

[13] Abhijit Guha, Abdulrahman Alahmadi, Debabrata Samanta, Mohammad Zubair Khan, and Ahmed H. Alahmadi. A Multi-Modal Approach to Digital Document Stream Segmentation for Title Insurance Domain. *IEEE Access*, 10:11341–11353, 2022. 2 citations (Crossref) [2022-08-21] Conference Name: IEEE Access.

[14] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Peter A. Hancock and Najmedin Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183. North-Holland, 1988.

[15] Camille Jandot, Patrice Simard, Max Chickering, David Grangier, and Jina Suh. Interactive semantic featuring for text classification. *arXiv preprint arXiv:1606.07545*, 2016.

[16] Twin Karmakharm, Nikolaos Aletras, and Kalina Bontcheva. Journalist-in-the-loop: Continuous learning as a service for rumour analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 115–120, Hong Kong, China, November 2019. Association for Computational Linguistics.

[17] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[19] Antonio Mucherino, Petraq J. Papajorgji, and Panos M. Pardalos. *k-Nearest Neighbor Classification*, pages 83–106. Springer New York, New York, NY, 2009.

[20] Anh Khoi Ngo Ho, Véronique Eglin, Nicolas Ragot, and Jean-Yves Ramel. A multi-one-class dynamic classifier for adaptive digitization of document streams. *International Journal on Document Analysis and Recognition (IJDAR)*, 20(3):137–154, September 2017. 1 citations (Crossref) [2022-10-24].

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[22] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[23] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520, 10 2004.

[24] Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[25] Christian Shin, David Doermann, and Azriel Rosenfeld. Classification of document pages using structure-based features. *International Journal on Document Analysis and Recognition*, 3(4):232–247, May 2001.

[26] Patrice Simard, David Chickering, Aparna Lakshmiratan, Denis Charles, Leon Bottou, Carlos Garcia Jurado Suarez, David Grangier, Saleema Amershi, Johan Verwey, and Jina Suh. Ice: Enabling non-experts to build models interactively for large-scale lopsided problems, 2014.

[27] Alison Smith, Varun Kumar, Jordan Boyd-Graber, Kevin Seppi, and Leah Findlater. Closing the loop: User-centered design and evaluation of a human-in-the-loop topic modeling system. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, page 293–304, New York, NY, USA, 2018. Association for Computing Machinery.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[29] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A Survey of Human-in-the-loop for Machine Learning. *arXiv:2108.00941 [cs]*, November 2021. arXiv: 2108.00941.

[30] Shuo Xu. Bayesian naïve bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59, 2018.

[31] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. *arXiv:2007.14062 [cs, stat]*, January 2021. arXiv: 2007.14062.

[32] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1:43–52, 12 2010.

Appendix A

MODEL SETUP

This section shows the setup of the models used in the experiments.

Models	**Setup**
Support Vector Machine	SVC(C=15, decision_function_shape='ovo', kernel='linear')
Random Forest	RandomForestClassifier(n_estimators=20, random_state=42)
XGBoost	XGBClassifier()
Naive Bayes	GaussianNB()
TFIDF	TfidfVectorizer(max_features = 10000)
SBERT	model = SentenceTransformer('sentence-transformers/bert-base-nli-mean-tokens')
doc2vec	model = Doc2Vec(documents, vector_size=50, window=2, min_count=1, workers=4)
BERT	Batch Size = 8, No. of Epochs = 20, Number of trainable Parameters = 109490699, attention_probs_dropout_prob": 0.1, hidden_act = "gelu", hidden_dropout_prob = 0.1, hidden_size = 768, Pretrained Model = bert-base-uncased
Longformer	Batch Size = 4, No. of Epochs = 10, Number of trainable Parameters = 148667915, attention_probs_dropout_prob": 0.1, hidden_act = "gelu", hidden_dropout_prob = 0.1, hidden_size = 768, Pretrained Model = allenai/longformer-base-4096

Table A.1: This table contains the setup of each model used in the experiments. The Setup column contains the required information about the model to replicate the results using the company's dataset.