Document Classification and Information Extraction framework for Insurance Applications

Ananth Raj GV, Qian You, Eric Bunch, James Kim, Marepally Santosh, Glenn Fung
American Family Insurance, Machine Learning Research Group
{agauribi, qyou, ebunch, jjkim, smarepal, gfung}@amfam.com

Abstract

Document Intelligence is an essential subclass in the field of machine learning. It plays a vital role in insurance applications and other sectors. In this work, we showcase a business application that uses two different but complimentary techniques: document classification and entity extraction. We also provide an overview of an end-to-end production level system that incorporates deep learning models deployed at scale. The system's backbone relies on trained models carefully analyzed and designed to generalize well on existing and future use-cases. Through empirical evidence, we provide insights into several models trained on our insurance-related datasets and highlight models that have shown good performance across multiple datasets in our real-world insurance setting.

1 Introduction

Every year, insurance companies consume millions of document images of hundreds of categories, including medical documents, checks, attorney correspondence letters, police reports, and subrogation documents. These documents contain critical information such as personal identifiers, claim numbers, policy identifiers, medical providers, and insured property information that communicate across business workflows and systems. Therefore, these documents primarily drive the insurance claim processing workflows and play essential roles in other business divisions. Being able to classify and extract relevant information from these documents automatically can significantly improve the efficiencies of numerous business workflows, reduce manual operation cost, enhance the quality and the re-usability of crucial information. We designed, implemented and deployed an automated document classification and information extraction pipeline (see Figure 1) that can democratize this functionality across the enterprise and could be used at scale for many similar use-cases.

Document image classification and information extraction machine learning approaches have been very well researched (Das et al. 2018; Kang et al. 2014; Katti et al. 2018; Chalkidis and Androutsopoulos 2017; Liu et al. 2019; Huang, Xu, and Yu 2015). However, most of the algorithms are developed based on only a few open-source data sets (Harley, Ufkes, and Derpanis) with limited types of general business documents. In our insurance context, document formats can range from entirely unstructured, semi-structured to fully structured, such as tables and forms, logos, letterheads, and handwritten scripts. Therefore in this paper we sort general document classification and information extraction methods that have potential to be generalized to current and future documents. Our proposed methods have three contributions, which we describe below.

The first contribution is the application of our document classification and information extraction pipeline to automate two relevant insurance workflow applications: a Medical Bill classification for claims, and a Salvage Claims application. In the medical bill application, we use deep learning models to first classify the type of medical bill, and then extract claimant information. These models will be used to improve a medical bill workflow that handles 300,000 and growing medical bill documents per year that are submitted by our customers to support during our claims process. In the auto mobile salvage claim application, we use document classification to identify bank-issued letters of guarantees for loan payment and extract car information. This automation will reduce manual operation cost and improve the processing efficiency of 100,000 total loss claims annually.

The second contribution is the creation of a simple yet effective character level sequence-to-sequence model for information extraction. Previous information extraction from document images work mostly considers itself as an extension from Named Entity Recognition (NER). Conditional Random Fields (CRF) have been widely used for NER tasks but proved to be insufficient in our insurance data set. Because the documents can be tabular or the sequential pattern is non existent, CRFs are not necessarily a natural fit. These observations motivated us to devise a character level model that successfully learns and predicts the patterns of the string of characters, therefore achieving significantly better results than our BiLSTM-CRF (Huang, Xu, and Yu 2015) baseline. For the document classification task we altered the ensemble VGG model in (Das et al. 2018) to include word vector representations of the text in the document, further improving

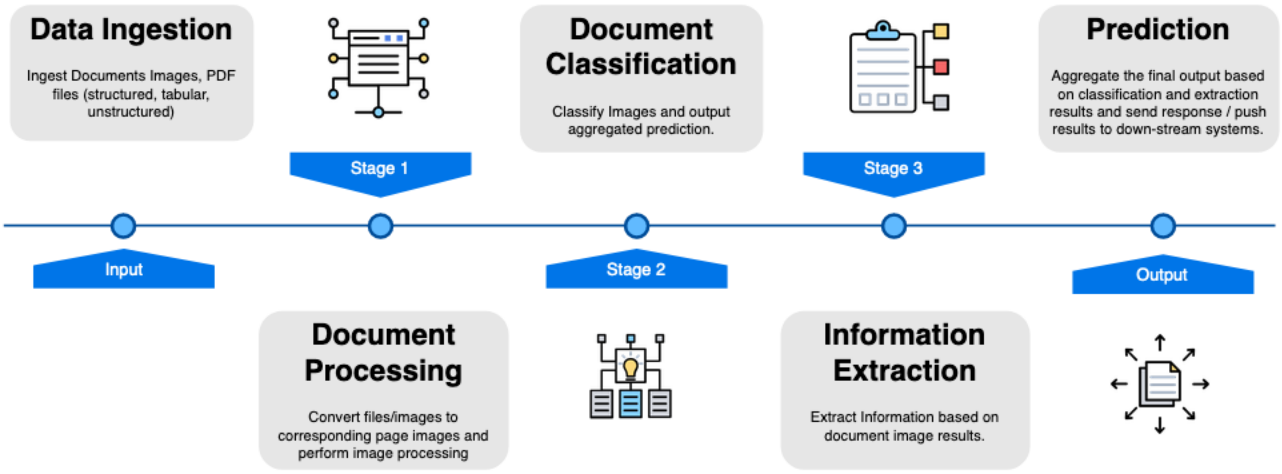


Figure 1: Document Image Classification and Information Extraction Pipeline

classification performance.

The third and the last contribution is the deployment of these models using a scalable architecture consisting of a Elastic Container Service (ECS) running on AWS Fargate. We also discussed how our deployment can be automatically scaled to address demand.

We develop a fast API-based rest application to consume PDF files from an upstream system. Up-steam systems currently post rest requests with PDF files as byte streams. The requests are subsequently processed in 3 stages, Figure 1 - **Document Processing**, **Document Classification** and **Information Extraction**. Document processing involves processing the document into page-level images. The second stage classifies the page images into class labels, which then moves to the third stage of the pipeline - the information extraction phase. This stage extracts relevant information based on the application. The pipeline's final output is the prediction of both the classification and extraction model based on the business rules.

2 Related work

Prominent deep-learning-based methods for PDF document image classification tend to be either image-based or text-based. The method described in (Das et al. 2018) uses an ensemble of VGG16 models that have been pretrained on ImageNet to classify documents from the RVL-CDIP data set (Harley, Ufkes, and Derpanis). The ensemble consists of five separate pretrained VGG16 models, each trained on the RVL-CDIP data set using either a left, right, top, bottom, or no crop of the image, and achieves an accuracy of 92.2%.

Information extraction from documents has traditionally been treated as a text-based task, using model architectures such as BILSTM-LSTM and BILSTM-CRF (Chalkidis and Androutsopoulos 2017). Techniques that further incorporate structural information of the documents include graph convolutional networks (Liu et al. 2019), which encodes the document structure as a graph with regions of text as nodes and edges encoding information such as aspect ratios and distance between the regions being connected. The model employs a method akin to graph convolution before applying a BILSTM-CRF model. It should be noted that both the work in (Chalkidis and Androutsopoulos 2017) and (Liu et al. 2019) operate on partial word tokens, whereas the setup of the proposed work necessitates a character level model. This is due to the fact the documents concerned have significantly less structure. It should be noted that the documents used in (Chalkidis and Androutsopoulos 2017) are relatively noiseless, and have enough structure to allow for the use of heuristics to pre-process and segment the document.

Chargrid (Katti et al. 2018) and BERTgrid (Denk and Reisswig 2019) are two techniques that encode the spatial structure of the documents as 2D grids of characters and contextualized token vectors respectively. Chargrid uses a one-hot encoded vector to represent the characters, whereas BERTgrid represents partial word tokens using contextualized vectors from the popular BERT model (Devlin et al. 2019). Both models then employ a convolution encoder-decoder architecture that predicts bounding box coordinates as well as the character or word token within the box. In a similar vein to these two methods, there has been work using a Mask R-CNN architecture, called PubLayNet (Zhong, Tang, and Yepes 2019), to detect segments of documents(e.g. table, paragraph, figure), using a large annotated document data set.

LayoutLM (Xu et al. 2020) is a pre-training method that combines the text of the document with the layout of the 2D document, also employing BERT representations of word tokens, as well as an image embedding for bounding boxes around identified words. These learned representations can be used for downstream information extraction and document classification tasks, and has been shown to outperform many previous SOTA benchmarks in this domain.

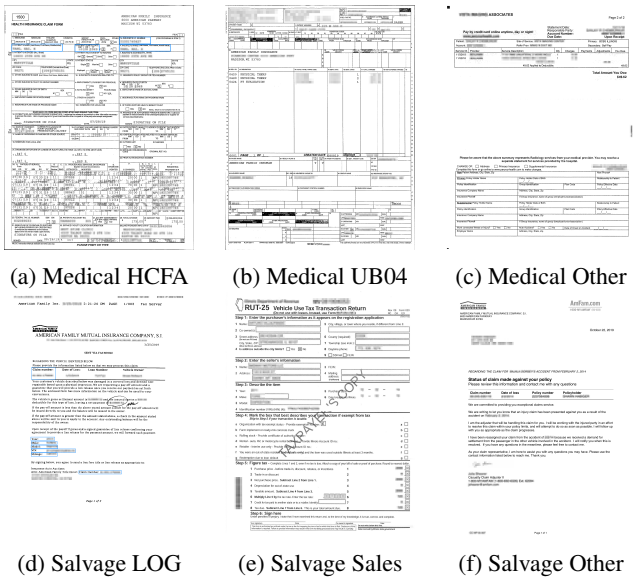
3 Applications

The Medical Bills application

Our insurance company has been ingesting medical bills from external health institutions for automobile insurance casualty claims review and re-pricing. Streaming in from snail mails and emails throughout the day, medical bills consist of three types: HCFA, UB, and Non-standard bills. Figure 2(a)-(c) shows a sample image of each type of medical bills. All medical bills are mostly in a tabular format, with HCFA and UB having type-specific layout. Other medical bills table structures can be arbitrary. Currently, a team of operational support staff manually classify medical bills into three classes and extract relevant content such as claim identifiers, date of birth (DOB), names, addresses, patient identifiers using OCR or other template matching algorithms. Another team of operational support staff is responsible for auditing the classification and extraction results. The classified documents and extracted information of the medical bill will then be stored and reused. Approximately 300,000 medical bills need to be processed per year, and downstream claim processing workflows will reuse the classified and extracted information. This paper intends to propose generic machine learning approaches to automate both medical bill classification and information extraction. Our proposed method can significantly reduce labor cost, task turn-around time, and overcome the limitations of brittle template-matching algorithms that will fail in the case of new document formats. The growing volume of medical bills also motivates us to deploy our method on the cloud to meet scalability needs and to democratize this utility to other business functions.

The Salvage Claims application

Personal Automobile Insurance line processes 80,000 to 100,000 total loss claims per year. A total loss automobile claim, or a salvage claim, is filed when the damaged automobile's repair cost or salvage cost exceeds the insured value. One typical workflow to process a salvage claim is to collect a letter of guarantee (LOG) issued from banks that previously issued lease or loan for the damaged automobile. LOG guarantees that the insurer's reimbursement will pay the bank first and then the automobile titleholder. Information such as claim id, vehicle identification number(VIN), year, make, and car model will be extracted from the LOG to complete the salvage claim processing. We typically receive 3500 to 4000 salvage claims email per month, with 40 documents per hour at peak time. The email attachments consist of LOG, Sales Receipt, and other types of documents (Figure 2(d)-(e)). The LOG content is usually semi-structured with a block containing claim id, vin number, year, make, and car model. Sales receipt and other documentation associated with salvage claims can be both tabular or unstructured. Few assumptions can be made on the position where the relevant claim and car information will appear. One to two full-time employees are responsible for classifying documents attached to the email and extract relevant information. The manual process is quite repetitive and expensive, and the salvage claims usually involve low monetary values and need little investigations. Therefore we also intend



(a) Medical HCFA (b) Medical UB04 (c) Medical Other

(d) Salvage LOG (e) Salvage Sales (f) Salvage Other

Figure 2: Samples of (a) HCFA Medical Bill (b) UB Medical Bill (c) Other Medical Bill, (d) Salvage Claim Letter of Guarantee (e) Salvage Claim Sales Receipt (f) Other Salvage Claim document

to use our proposed approach to eliminate human labor in this particular use case to achieve the long term goal of completely automating thousands of claims a year. The continuously incoming emails require near real-time capability, which motivates us to design different scaling points when deploying our models as an API on the cloud.

Labels And Annotations

We adopt a supervised learning approach for both document image classification and information extraction. We enhance an open source labeling tool Prodigy (Neves and Ševa 2019) to label document classes and to annotate bounding boxes around claim id, name, addresses, DOB and other information(Figure 3). OCR tool such as pytesseract (Hoffstaetter, Bochi, and Lee 2014) is then used to extract character information from the bounding boxes.

4 Methods

Document Image Classification

We describe three models and how we alter them to achieve high accuracy for both the applications we are trying to solve. We start with the baseline VGG model (Simonyan and Zisserman 2014) and compare the accuracy with modified architectures Figure 4. The baseline VGG model performs well on classification tasks, but recent new ensemble models that we use have proved to provide better accuracy scores than the baseline model.

VGG Baseline VGG16 (Simonyan and Zisserman 2014) is a popular class of Deep Learning Convolutional Neural Networks that have been widely used for Image Classification. VGG16 consists of 16 layers of a combination of

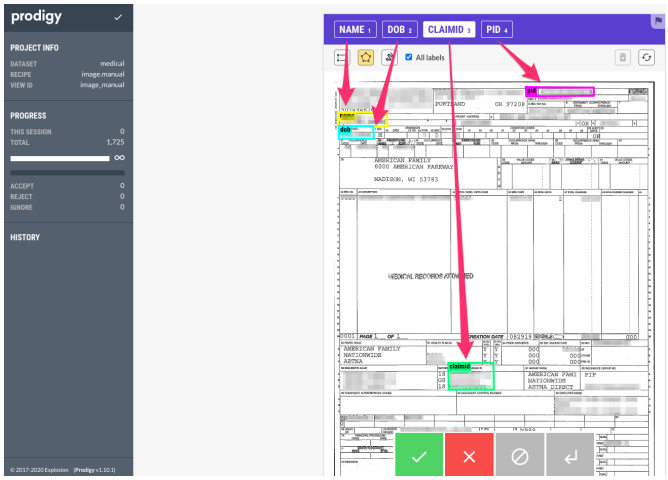


Figure 3: Prodigy as a tool for labeling

convolutional, max pooling, fully connected layers, and a 1000-way softmax classifier. This model achieves an accuracy of 92.7% on the ImageNet dataset. The dataset comprises 14 million images belonging to 1000 classes. We use this model pre-trained on ImageNet and train it on our custom dataset using the concept of transfer learning. The input to the model are images resized to (256, 256) in shape. We freeze the weights of all the layers except the classification layer for our baseline model during training and change the last classification layer of the model to output the softmax scores over three classes(LOG, SALES, OTHER) for Salvage and (HCFA, UB 04, OTHER) for Medical application.

Ensemble VGG VGG models can be utilized in other ways to classify images. Recent methodologies have introduced the concept of intradomain transfer learning. The idea is to train multiple VGG models for specific regions and use them as feature extractors for an ensemble model (Das et al. 2018). We train five such region-specific models targeting specific regions - top, bottom, left, right, and holistic, respectively. We first train a base VGG model on the RVL dataset (Harley, Ufkes, and Derpanis) consisting of 400,000 grayscale images in 16 classes. The trained model is then used as the starting point for transfer learning to extract region-specific features for our application. We resize the images from our dataset to (256, 512) dimensions for top and bottom regions and (512, 256) dimensions for the left and right regions. After fine-tuning these models, we derive five different models. They are used as feature extractors to our final ensemble model, which is a couple of fully connected layers and a softmax layer. Input to the ensemble model is horizontally stacked features from all the region models with a 3-way softmax classifier as the output.

Ensemble VGG + Text Features The final model we trained and currently use in production incorporates text-level features along with image features. This model is sensible because many insurance domain documents are distinctly similar at a template level. Hence, it is sensible to infer that incorporating text features when training a model would boost performance as the model attends to an image's visual characteristics coupled with text-level information. We use the same architecture as the Ensemble VGG classifier. However, a small change incorporates extracting a 300-dimensional document text embedding and appending it to the 1000 dimensional VGG features before passing it through Linear layers and a Softmax layer Figure 4. This enables the VGG model to adjust weights based on the text and visual features for each region. The output of the trained region models is stacked together as input to the following ensemble model. We observe better accuracy scores Table 1 and use this model in our production environment.

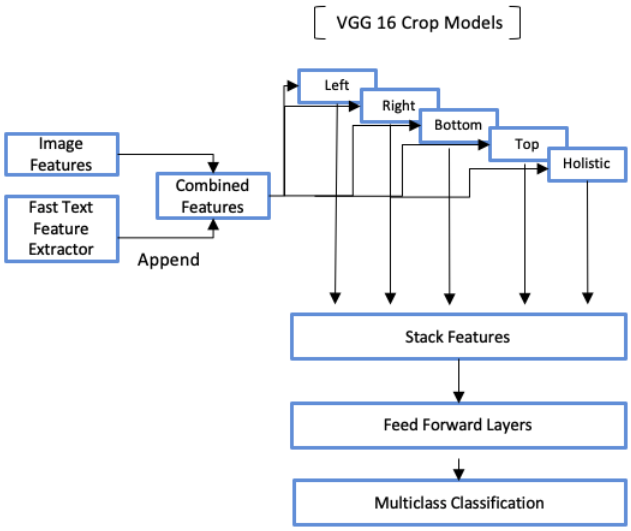


Figure 4: Ensemble VGG + Text Features Classification Model

Information Extraction

Information Extraction for document images is a widely sought out area in the domain of machine learning. In particular, insurance industries are researching and developing many models that can be generalized easily across multiple use cases. The main problem of information extraction is generalizing the model to have consistent performance across various templates(articles, tabular, diagrams, and numerical data). In general, documents have various templates, which makes developing an end-to-end model to extract information a difficult task. We use a Bidirectional LSTM-CRF model as our baseline and discuss some limitations of this method. Finally, we show how a Character-Based BiLSTM model addresses those issues by generalizing well irrespective of the dataset and show improved accuracy scores in our application.

NER BiLSTM-CRF Baseline Named Entity Recognition (NER) is one of the most widely used information extraction techniques. With the advent of LSTM networks in deep learning, NER problems have been tackled using LSTMs, BiLSTMs, and BiLSTM-CRFs (Figure 5). The CRF layer has proven to be a useful tool in the NER task domain, due

to its ability to model semantic structure of the ordering of tokens present in the text. However, when modeling documents like semi-standardized forms, there is not much semantic structure present in the word tokens (as opposed to say a wordy legal contract, as in (Chalkidis and Androutsopoulos 2017)).

The CRF model's basic intuition is that the LSTM layer attends to past input features, and the CRF layer utilizes sentence level tag information. A noticeable limitation of the CRF model is that it is designed to detect token transitions in a long text sequence. It is not very effective in distinguishing the representations of the token itself.

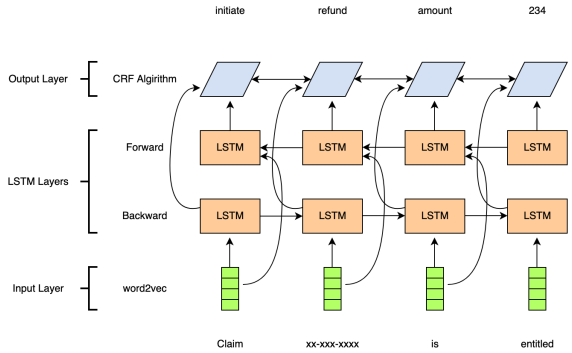


Figure 5: NER BiLSTM-CRF Baseline Model

Character Based Sequence Classification To overcome the limitations of the CRF model, we use a character-based BiLSTM sequence model. BiLSTM networks can be extended to a character level (Gridach 2017) where the input to the model is character sequence and not individual tokens. In our use case, examples of the Claim number and VIN could be 1234-56789-123 and JPAG764589 respectively. Framing this as a character level sequence to sequence model, the model tends to learn the characters' internal representation in tokens. Input to the model are character encodings, and the output is processed to capture the longest subsequence of class labels for all predicted fields. Figure 6 shows a hypothetical output of the model: 0001111002222000333. Here, 1 would represent the claim id subsequence, 2 would represent VIN, and so forth. In this workflow, OCR is used to extract text from the document. This process is not perfect, and can result in incorrectly extracted text (e.g. a "1" instead of an "I"). One benefit of a BiLSTM model trained on character sequences is that it generalizes well on the dataset despite these errors in the OCR module. We use ASCII characters, numbers, and special characters as our vocabulary for our embeddings. We then train the model in a multi-class classification setting. We use a softmax layer to output the probability distribution across six classes for both medical bills and salvage claims. We then classify the sequence based on the maximum probability scores.

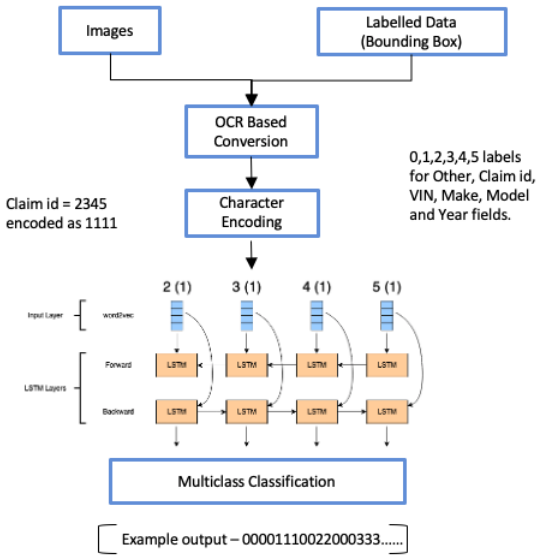


Figure 6: Character Based Sequence Classification Model

Data	Model	Auc(ovr)	Auc(ovo)	Acc
Salvage	VGG Baseline	0.977	0.976	94.72
	ENS VGG	0.998	0.998	97.47
	ENS VGG + TF	**0.999**	**0.999**	**99.26**
Medical	VGG Baseline	0.959	0.964	82.41
	ENS VGG	0.978	0.979	86.66
	ENS VGG + TF	**0.999**	**0.999**	**98.13**

Table 1: Accuracy and AUC Scores for the classification models, VGG, ENS VGG (Ensemble VGG), and ENS VGG + TF (Ensemble VGG with Text Features). The abbreviations ovr and ovo denote "one-vs-rest", and "one-vs-one", respectively.

5 Experiments and results

Datasets and Experiment Setups

For the Medical Bills application, we collect 1500 Medical bills consisting of 500 examples from each of the 3 classes - HCFA, UB04 and Non-standard bills, along with their class labels. We labelled and annotate 700 medical documents for extracting the relevant fields. For the Salvage Claims application, we collect a dataset that consists of 2,000 documents each of Letter of Guarantee, Sales Tax, and Other class. We labelled and annotate 700 medical documents for extracting the relevant fields. Both the applications first classify these documents and then extract information: claim num-

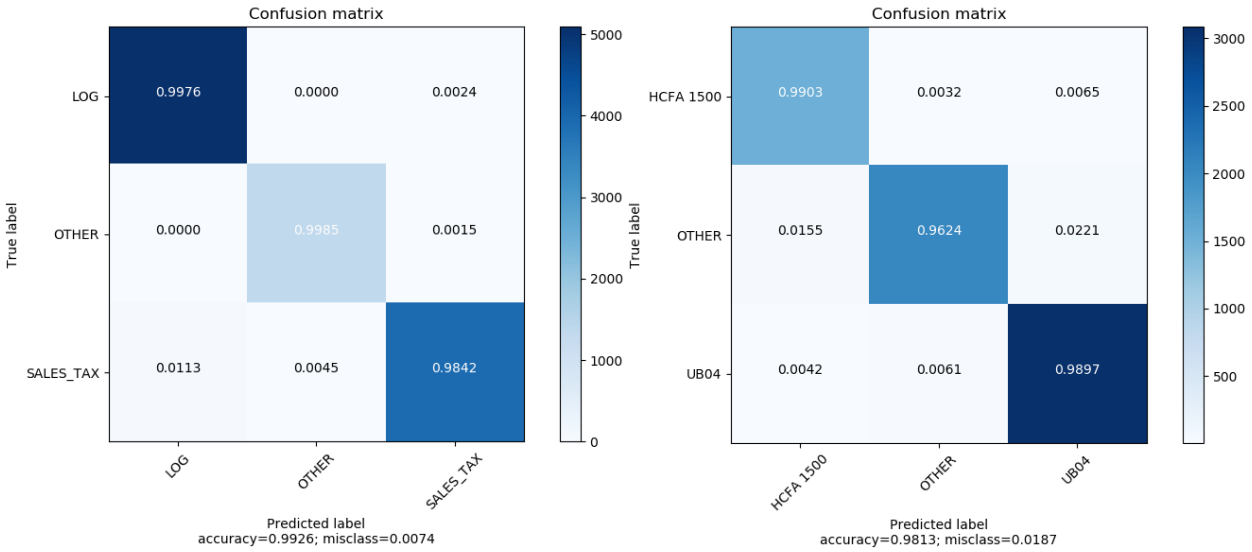


Figure 7: Confusion Matrices of ENS VGG + TF for Salvage Claims (left) and Medical Bills (right) applications.

Application	**Field**	**Precision**		**Recall**		**F1**	
		CRF	**Char**	**CRF**	**Char**	**CRF**	**Char**
Salvage	OTHER	1.00	1.00	1.00	1.00	1.00	1.00
	CLAIM ID	0.30	**0.89**	0.38	**0.96**	0.33	**0.92**
	VIN	0.45	**0.72**	**0.71**	0.65	0.56	**0.68**
	MAKE	0.17	**0.57**	0.20	**0.64**	0.18	**0.60**
	MODEL	**1.00**	0.67	0.50	**0.61**	**0.67**	0.64
	YEAR	0.50	**0.51**	0.50	**0.57**	0.50	**0.54**
Medical	OTHER	**1.00**	0.96	**1.00**	0.97	**1.00**	0.96
	NAME	0.18	**0.53**	0.22	**0.90**	0.20	**0.67**
	DOB	**1.00**	0.65	0.33	**0.89**	0.50	**0.75**
	CLAIM ID	0.33	**0.50**	0.29	**0.93**	0.31	**0.65**
	PID	0.40	0.40	0.36	**0.91**	0.38	**0.55**

Table 2: Information Extraction Precision, Recall, and F1 scores for CRF (BiLSTM-CRF Baseline) and Char (Character Based BiLSTM).

ber, name, DOB, and patient id for medical bills; claim number, VIN, make, model, and year for salvage claims. The annotated bounding box coordinates are processed as input to the extraction model. We train the classification model and extraction model on p3.2x large GPU instances on AWS.

Document Pre-processing

Documents are generally in PDF format, and are converted to respective page-level images using the python package pdf2image. We enhance the images to 600 dpi, convert them to grayscale, and retain the images' original dimensions. The enhancement in the quality of images facilitates better OCR based image to text conversion and good feature learning during training. Images are then processed using PyTesseract OCR (Hoffstaetter, Bochi, and Lee 2014) to extract text from images.

To train our classification and extraction model, labeled images are divided into training, validation, and test data. We follow an 80%, 10%, 10% split in our data sets. We incrementally report our results, starting with base models towards models with better performance.

Document Classification result

The evaluation metrics used for the classification model are accuracy and AUC score. Since this is a classic multi-class classification problem, we compute the AUC of each class against the rest, also called OVR (Pedregosa et al. 2011), and compute the average AUC of all possible pairwise combinations of classes, also called OVO (Pedregosa et al. 2011). From Table 1, we can see that the ensemble VGG model trained on region features and text features (combined) outperforms the models trained otherwise and generalizes well

across both medical and salvage applications. We depict a 3% increase in accuracy and a substantial improvement in AUC scores in our final model compared to the Salvage dataset's baseline model. There is a considerable improvement in accuracy and AUC scores for medical bills as well. Medical bills being highly structured, we can conclude that text-level features play an important role in model classification. We also report the confusion matrix for our classification model as shown in Figure 7. It is evident that model makes negligible misclassifications in both the applications.

Information Extraction result

The evaluation metrics used for the extraction model are precision, recall, and F1 scores. We report field level metrics to get a detailed view of how the model performs in extracting information. Reporting these metrics to the business enables them to decide the trade-off in precision, recall, and set a threshold to loop in a human for low confidence results.

NER BiLSTM-CRF Baseline Based on our experiments on our insurance dataset, we observe that, though BiLSTM-CRF is the go-to option for Information Extraction, these models may not always perform well on all datasets. Our dataset is a combination of articles, tabular, and data of other forms where the tokens to be classified are not a sequence of tokens but relatively sparse words in a long text sequence. Textual information is derived from OCR processing, so there is always a setback in our pipeline that PDF documents have probable low-quality scans. Hence text derived from OCR processing is often partial, and tokens to be categorized could have different rendering patterns based on tabular or article documents.We observe low precision and recall scores for our dataset under this methodology, Table 2.

Character Level Bidirectional LSTM Sequence Classification This model tends to learn a better representation among fields - name, DOB, claim number, PID in the medical application, and claim id, year, model, make, and VIN in the salvage application. Unlike the BiLSTM-CRF model, we can conclude that the Character-based sequence classification model generalizes well on both medical and salvage applications. The model performs well in identifying and extracting fields because it learns the positional occurrence of specific fields like claim number with respect to others, such as make, model, or VIN during training. It also understands that specific fields are only numeric, only characters or alpha-numeric. This helps the model generalize well irrespective of the structure of the dataset. We observe good precision and recall scores for our dataset under this methodology, Table 2. Precision, recall, and F1 scores are high for claim id, VIN, make, model, and year in the salvage application and name, DOB, claim number and PID in the medical application compared to the baseline BiLSTM-CRF model. Let us take an example to understand why there is a boost in precision and recall scores for make, model, and year fields with regards to the character-based BiLSTM sequence model. For a particular salvage based LOG document, we see the occurrence of these fields as Subaru/LTX/2007. This corresponds to field entries - Make/Model/Year. When we use character-based encoding, the positional information of each character is taken into consideration and the character representations are learnt by the model. So the model understands that year appears after model and is usually a four character-based numerical entity having values between 0 and 9. Similarly, for instances where these fields occur as separate entities in any part of the document, there is a generalization principle learnt by the model.

6 Deployment

Container-as-a-Service deployment

We deployed and served classification and information extraction models in Amazon Web Service. We designed the production serving architecture by considering the incoming traffic frequency, volume, latency, and computing resources (e.g., CPU, GPU, RAM) for processing documents, and the business requirements.

Both the medical Bill and salvage applications have the requirement of near real-time processing. Documents are streamed in during the day with peak traffic of 40 documents per hour in the salvage application and an estimated number of daily documents of around 1000 in the medical bill application. It usually takes 20 to 30 seconds to process a one page PDF document. However, it can take up to 1 to 2 minutes when the incoming PDF document contains multiple pages. During processing, the classification stage needs to load large models such as FastText (Joulin et al. 2016) into memory for computing, and the total RAM used can exceed 6 GB. Due to the processing time and RAM requirements, we opted to run our document image classification and information extraction as an Elastic Container Service (ECS) task running on AWS Fargate rather than using a lightweight serverless architecture (AWS Lambda functions). The ECS tasks are configured with a computing capacity of 30GB RAM and 4 vCPU's to handle the requirements for processing each document.

Additionally, we use Elastic File System (EFS) to persist and load weights of any large models for inference. AWS Fargate tasks using EFS will mount the file system specified in the task definition and make them available to the containers. This enables persistent, shared storage to be defined and used at the task and container level in ECS, unlike the ephemeral container storage, which will be lost when the task stops. Shared storage can prevent duplicating the weights on every instance of the task. We download the 6.8GB weights of the fastText used in inference to the mounted EFS storage during the first task's initialization, which can subsequently be used by other tasks.

Two-phase deployment

In order to iterate fast in deployment, we executed the deployment into two phases. Figure 8(a) shows the phase I deployment using ECS running on AWS Fargate. We added a fast API to our model, dockerized it, and deployed it using the Jenkins pipeline. We plan to learn more about the latency of the API calls and this architecture's behavior under real traffic in the upcoming months. If the blocking HTTP calls to this API can not meet the latency requirement due

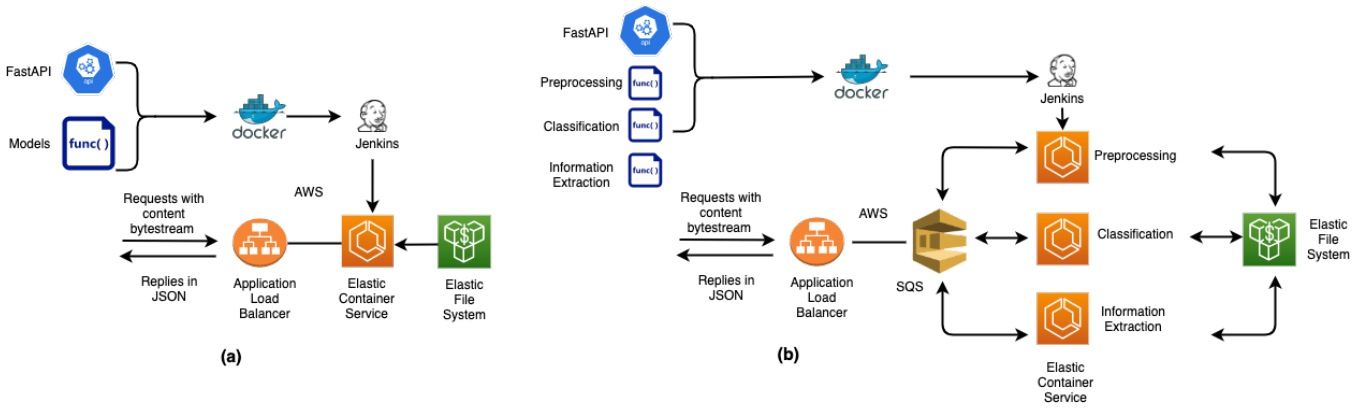


Figure 8: Deployment Phases (a) ECS running on AWS Fargate (b) Independent scaling of pre-processing, classification and extraction stages with Simple Queue Service

to document size or any traffic growth, we will switch to a phase II deployment by adding a few scaling elements to the phase I deployment architecture. Figure 8(b) shows how we can further scale up the architecture by decomposing the models computing logic into three parts and set up three discrete ECS tasks on AWS Fargate to scale independently: pre-processing (primarily converting pdf to images), classification, and information extraction. A queuing system such as Simple Queue Service (SQS) could handle the communication between these three steps.

7 Conclusion and future work

Automatic document image classification and information extraction (DOCIE) are two frequently occurring tasks in enterprises such as insurance companies. Millions of documents need to be continuously organized and indexed on an on-going basis. In this paper, we proposed an end-to-end system that accomplish this chain of tasks by leveraging both state-of-the-art computer visions and language models. We have introduced two innovative and impactful applications for document image classification and information extraction: medical bill classification and claimant information extraction for casualty lines, and the other is classifying a type of contract issued by banks and car information extraction for automobile salvage claims. As many more types of document images from various business functions will be automated using our system, our algorithms' design attempts to address the various formats of document images encountered in our current and future applications. As seen from our results, the extraction model we use being a Character-based BiLSTM Sequence model, they can generalize well irrespective of the structure of the incoming documents. The model may hence do a good job addressing any business use case in the foreseeable future. The extracted information could be stored as metadata of each document that is streamed into the company and be reused by many downstream processes. The design of our two-phase deployment accounts for horizontal and vertical scalability. Further, low latency for real-time applications can be achieved by optimizing the combination of CPU and GPU usage at different points of computing e.g., paralleling OCR using multiple processes and running machine learning models on GPU. We are also exploring some of the newly proposed information extraction methods, such as LayoutLM, that leverages layout information. We plan to generalize our models to extract comprehensive types of personal identification information and entities of interest from a wide range of insurance documents, including personal checks, purchase receipts, tax documents, medical records, subrogation letter, etc. Building large pre-trained data sets using larger sets of insurance-specific data sets can also be beneficial. We can use transfer learning, few shot learning, or active learning to reduce labeling efforts for specific applications.

References

Chalkidis, I., and Androutsopoulos, I. 2017. A deep learning approach to contract element extraction. In *JURIX*.

Das, A.; Roy, S.; Bhattacharya, U.; and Parui, S. K. 2018. Document image classification with intra-domain transfer learning and stacked generalization of deep convolutional neural networks. *2018 24th International Conference on Pattern Recognition (ICPR)*.

Denk, T. I., and Reisswig, C. 2019. Bertgrid: Contextualized embedding for 2d document representation and understanding.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. *Proceedings of the 2019 Conference of the North*.

Gridach, M. 2017. Character-level neural network for biomedical named entity recognition.

Harley, A. W.; Ufkes, A.; and Derpanis, K. G. Evaluation of deep convolutional nets for document image classification and retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Hoffstaetter, S.; Bochi, J.; and Lee, M. 2014. pytesseract.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging.

Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; and Mikolov, T. 2016. Fasttext.zip: Compressing text classification models. *CoRR* abs/1612.03651.

Kang, L.; Kumar, J.; Ye, P.; Li, Y.; and Doermann, D. S. 2014. Convolutional neural networks for document image classification. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, 3168–3172. IEEE Computer Society.

Katti, A. R.; Reisswig, C.; Guder, C.; Brarda, S.; Bickel, S.; Höhne, J.; and Faddoul, J. B. 2018. Chargrid: Towards understanding 2d documents. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Liu, X.; Gao, F.; Zhang, Q.; and Zhao, H. 2019. Graph convolution for multimodal information extraction from visually rich documents. In *NAACL-HLT*.

Neves, M., and Ševa, J. 2019. An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*. bbz130.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition.

Xu, Y.; Li, M.; Cui, L.; Huang, S.; Wei, F.; and Zhou, M. 2020. Layoutlm: Pre-training of text and layout for document image understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhong, X.; Tang, J.; and Yepes, A. J. 2019. Publaynet: largest dataset ever for document layout analysis.