Harvest - a System for Creating Structured Rate Filing Data from Filing PDFs

Ender Tekin[1], **Qian You**[2*], **Devin M. Conathan**[1], **Glenn M. Fung**[1], **Thomas S. Kneubuehl**[1]

[1] American Family Mutual Insurance Co. S.I.
[2] Coupang Corp.
etekin@amfam.com, qyou@coupang.com, dconatha@amfam.com, gfung@amfam.com, tkneubue@amfam.com

Abstract

We present a machine-learning-guided process that can efficiently extract factor tables from unstructured rate filing documents. Our approach combines multiple deep-learning-based models that work in tandem to create structured representations of tabular data present in unstructured documents such as pdf files. This process combines CNN's to detect tables, language-based models to extract table metadata and conventional computer vision techniques to improve the accuracy of tabular data on the machine-learning side. The extracted tabular data is validated through an intuitive user interface. This process, which we call Harvest, significantly reduces the time needed to extract tabular information from PDF files, enabling analysis of such data at a speed and scale that was previously unattainable.

Introduction

Tabular data frequently appears in the vast amounts of documents that need to be processed in many insurance workflows. However, these documents are usually unstructured, appearing either in the form of scanned documents or images, as well as documents that were born digital, but lack a structured representation of the data due to limitations of document types such as PDF files. As such, any process that can efficiently extract tabular data and convert it into structured format can lead to significant savings in processing time and cost. In this paper, we focus on this problem, in particular as it pertains to extracting tables from insurance companies' public rate filing documents.

To price a customer for a policy, insurance companies use factor books, sets of tables that provide different weightings to factors such as customer's age, history, vehicle/property type etc. These factors are filed by each company to each state's regulatory agency (such as a Department of Insurance), and are public. Historically, insurance companies have used these factors to benchmark their pricing. However, obtaining this information from filing documents is in general a manual, cumbersome and time-consuming process, limiting the ability of companies to analyze this information and act accordingly. Doing this in an efficient and affordable manner can provide a large competitive advantage, especially when it comes to bringing in new customers.

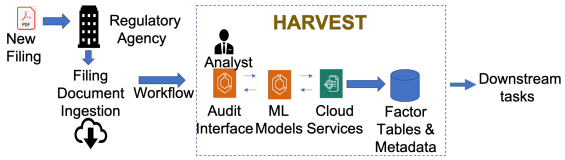

Figure 1: Harvest process

In this paper, we report on a process called *Harvest* that we developed to extract tabular data from rate filings in an efficient and scalable manner. This cloud-based process deploys a combination of machine learning algorithms in a way that allows processing large filing documents in a scalable manner, and provides a validation user interface that ensures that the extracted structured tables accurately represent the data contained in the original documents. Harvest can significantly cut down the time to process a filing, allowing an analyst to focus on analyzing the data and setting strategy rather than the tedious task of manually extracting the information. Furthermore, the extracted information is then available to be consumed by a variety of downstream tasks such as analyses for marketing/sales, strategy or pricing. This process is summarized in Figure 1.

Table Detection and Recognition

General document analysis, and specifically extracting tables from unstructured documents has been the subject of ongoing research for almost as long as digital documents have been around. Competitions such as ICDAR 2013[†] & 2019 (Gao et al. 2019) have recognized the importance of table detection and recognition and have sought to further the interest in unstructured document understanding. Datasets from these competitions, along with others such as Marmot[§] and TableBank (Li et al. 2019) have been released to support research in this area.

Traditionally, table detection has been approached using classical computer-vision techniques: finding lines/borders via edge detection techniques and then combining them to

*The author's contributions to this paper were produced while the author was employed at American Family Insurance.

[†]https://www.tamirhassan.com/html/competition.html

[§]http://www.icst.pku.edu.cn/cpdp/data/marmot_data.htm

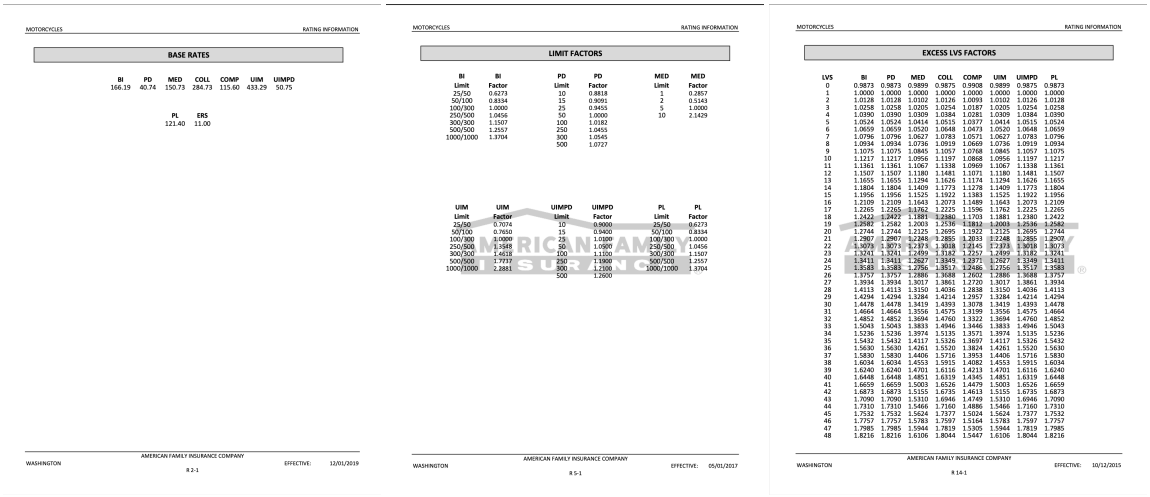

Figure 2: Rate filing pdfs. Tables can be affected by watermarks, as well as lack common table features such as border lines.

construct the tables (Cesarini et al. 2002; Kasar et al. 2013). As CNN's have exploded in popularity, more approaches that employ CNNs or other deep learning methods have taken over as the best performing approaches. For example, DeepDeSRT is another approach which uses a Fast-RCNN to detect the tables (Schreiber et al. 2017). This is followed by some heuristic processing to extract these tables. Paliwal et al. (2019) uses a bottom-up method to estimate table areas and table columns; two networks that share the same set of VGG19 features are trained to create semantic masks for each of these areas. These are then combined to heuristically and an OCR service, in this case Tesseract, is used to extract the tabular data. The Marmot dataset is used for training, with additional labeling of the columns to train the columns detection network. Kavasidis et al. (2019) uses a similar approach, classifying documents' salient parts in a pixel-wise fashion, followed by a fully connected CRF for localizing charts and tables.

Similar to Paliwal et al.'s work, (2019), we focus only on the table detection problem, and will rely on a third party service for the extraction of the data.

Rate Filings and Factor Tables

Insurance rate filings are mandatory filings by insurance companies to state regulatory agencies whenever there is a change in the factors that go into calculating insurance rates. These are publicly available either through state regulatory agencies themselves, or via third party data sources. These documents consist of a large number of factor rate tables, each of which can be many pages long. The format of these filings can change significantly from company to company, and the tables contained therein can also range from a single row to many tens of pages. Furthermore, they are contained in documents that contain a large amount of other information regarding the rate calculations, changes between filings, explanation of factors, etc. As a result, the filing documents can also range from tens of pages to over a thousand pages. The Harvest process aims to:

- Find all the tables corresponding to a user-provided list of factor table names. This is to avoid detecting the many superfluous tables that exist in documents, but are not of interest to the analysts.
- Extract the content of each table and provide them to the analyst for validation.
- Extract table metadata such as the specific names of the tables. A provided factor may consists of several different tables corresponding to that factor, e.g., a 'Base Rate' table may consist of 'Trailer Base Rate' and 'Safety Apparel Base Rate' tables, and both of these tables need to be detected and provided to the user.
- Allow the analyst to correct any errors in the tables and submit verified tables for storage in a cloud database that can be queried for various downstream tasks.

Example pages from a rate filings are shown in Figure 2. The filings we use for training and testing our models come from public filings of other insurance companies, or from within AmFam's other operating companies.

We note the lack of 'classic' table features such as borders as well as the potential existence of watermarks and the variety of shapes and structures of the tables that make traditional algorithms struggle with detecting such tables. From a performance perspective, the Harvest process needs to ensure that all relevant tables are detected and presented to the user, as missing tables in a filing would result in erroneous rate calculations. In contrast, from a user experience (analyst's) perspective, Harvest needs to ensure that there are not many false detections so as not to waste the user's efforts removing false detections from the end product. We discuss this in more detail in the User-Facing Results section.

Table Detection

The first step in Harvest is to determine which pages in these large documents that may have the tables the analyst is interested in. To find an initial set of candidates, the text from each page is extracted (via parsing the PDF file or OCR) and

the table names provided by the analysts are matched against the text in each page. Pages where the names of factor tables are found are then analyzed to see if any tables are present. This reduces the number of candidate pages by a significant amount, reducing the computational demands and cost.

We approach table detection as a general object detection problem, as was done by Schreiber et al. ((Schreiber et al. 2017)), and use a Faster-RCNN model to detect and localize tables in page images (Ren et al. 2015). Faster-RCNN is a state-of-the-art object detection framework that consists of a region proposal network that learns to identify potential objects, and outputs region proposals of these objects. These region proposals are classified as one of the objects of interest or background by a classifier network that is jointly trained with the region proposal network and shares the set of object features. This allows for faster training and inference, as the same backbone is used for both networks.

Model Training

To train the object detection models, we annotated several hundred pages of rate filing documents from several insurance companies' filings by drawing bounding boxes around the tables, captions and headers using the Prodi.gy annotation tool. These labeled datasets were split into training and test sets, ensuring that the images in the training and test sets come from separate filing documents from different states. This was done to ensure generalizability, since some tables span multiple pages, and having different pages of the same table in the test and training sets may give inflated accuracy results. Validation was performed using a five-fold process, where the training sets were randomly split into 5 subsets, and the training was done on 4 sets and tested on the fifth. This was done 5 times, each partition being used for testing once, and the average results across all 5 were used for parameter selection. In practice, training was seen to be fairly robust to parameters, so we only performed this to coarsely determine reasonable values for learning rate and a couple of other parameters. After that, we used the whole training set to train our Faster-RCNN models and test it on the test sets consisting of separate filing images.

Model Performance

COCO Metrics We used COCO metrics to evaluate the performance of our table detection models as it is a commonly used metric for object detection tasks (Lin et al. 2014). In general, very precise detection of boundaries is not necessary for extracting the factor rate tables. As such, we use the *mean Average Precision for IoU>0.5* as our precision metric, and *mean Average Recall @ 10* as our recall metric in our performance comparisons.

A base model was trained using the training sets for 4 companies for 20000 steps, with the learning rate being halved every 3000 steps. The results are shown in Table 3.

Looking at Figure 3, one might note the low recall values using the mean Average Recall @ 10 criterion. We believe that the main reason for the low recall values is due to the fact that the table detection models detect table-like structures within the actual tables marked as ground-truth, and

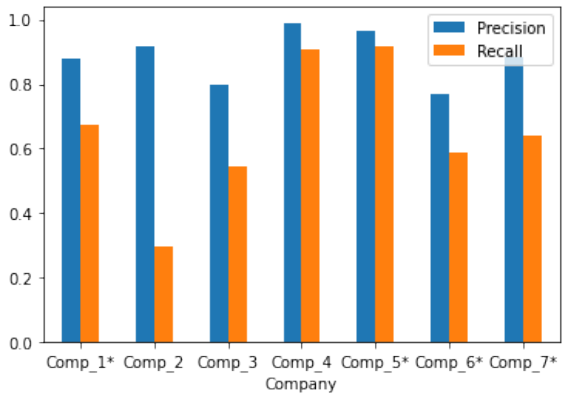

Figure 3: Base Model results. * indicates companies whose training sets were used for training this model. Results are shown on the distinct test sets for all companies.

these are considered as false negatives due to the IOU criterion used by COCO metrics. In production, we (1) merge the detected tables such that only the largest overlapping table is returned, and (2) consider confident detection of at least one table as the criterion for further processing, resulting in significantly improved recall values using the same model when the task is only detecting whether there is a relevant table in the page. From this perspective, COCO recall metrics are pessimistic, in that they are averaged over IoU values of [0.5, 0.95] with a step size of 0.05. We can calculate precision/recall metrics by varying the IoU threshold and confidence threshold for detection, we show one such example for Company 1 in Figure 4.

The variability of performance across different companies is especially notable, and for a performance-critical system as this, it is important that the performance meets certain criteria. To ensure that the model performs at the expected level across companies, it was determined that separate models be deployed for different companies, each model fine-tuned for that specific domain. This can be achieved by using labeled examples for each company to further refine the weights from the base model, potentially leading to significant improvements at the cost of increased labeling needs. To this end, we randomly sampled N=5, N=10, N=20, N=50, N=100 and N=200 samples from the training sets of each company and used these as the training sets, along with a matched number of examples from one of the companies in the original training sets. We performed this experiment 5 times, and show the average results in Figure 5.

We note that there is usually an immediate boost to performance with as little as 5 labeled examples for companies that were not in the original training set, as can be seen from the precision/recall plots for Company 3 (see Figure 6). This continues to improve until around 100 examples for precision before saturating, whereas for recall, the performance continues to improve. For companies that the baseline model was already performing well on, the improvement is limited, indicating that further fine-tuning is not necessary.

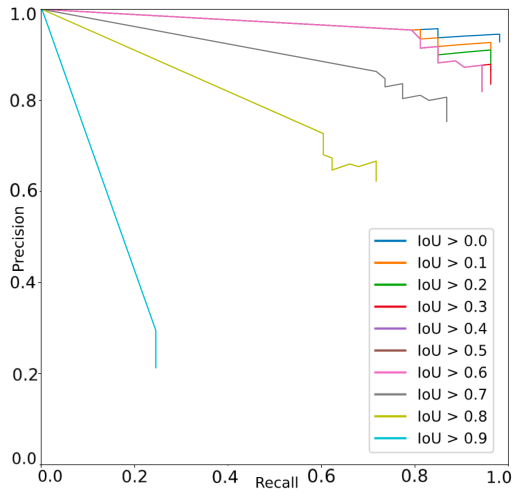

Figure 4: Table detection results for Company 1: the confidence threshold is varied for different IoU thresholds to get the various precision/recall plots. For our business case, it is important that the recall for this task is high, so we can set the thresholds to support this by choosing a combination of IoU and confidence thresholds. Note that even with a precise IoU threshold of 0.6 that allows accurate localization of tables, we can get around 0.9 precision & recall for this task.

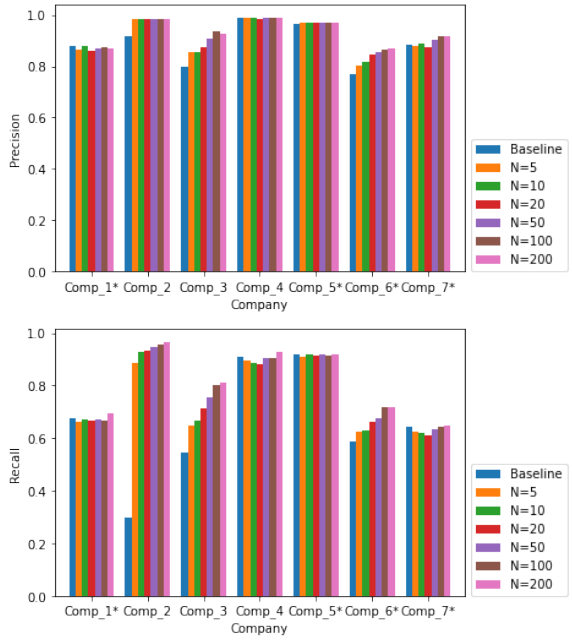

Figure 5: Effect of fine-tuning on precision (top) and recall (bottom) for various split sizes.

Observations We also explored using examples from other commonly used table datasets such as ICDAR 2013 and Marmot as extra training examples for our table detection models. However, we saw that this, in fact, reduced performance, further emphasizing that rate filing tables are visually dissimilar to commonly used tables in publications and similar documentation that form the bulk of table detection datasets. Many such datasets are collected from Word or LaTeXsource files, as it is easier to automatically create table datasets. However, this ends up skewing these datasets (and the models trained via these) to focus on conventional table features such as borders and the distinct look of LaTeX-produced tables in scientific publications.

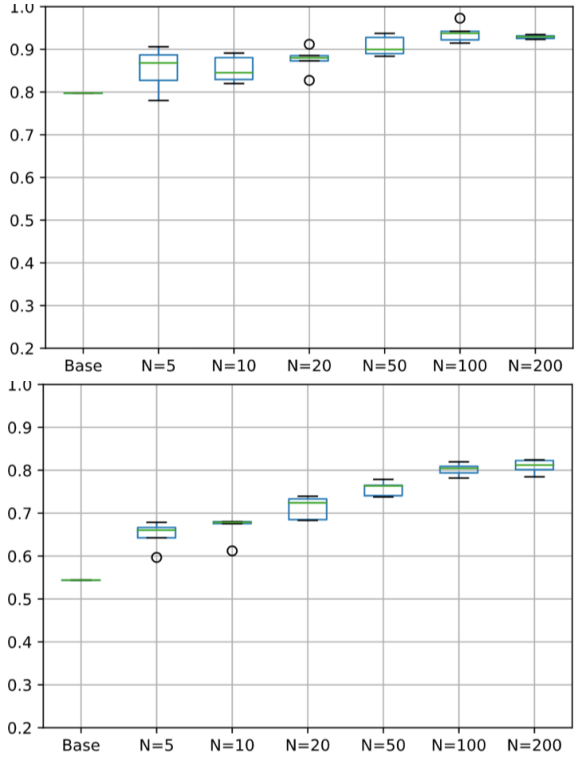

Figure 6: Fine-tuning results for company 3 vs the number of training examples. Precision on the top, recall on the bottom, results are averaged over 5 runs where the training examples were randomly sampled for each run. Note the initial improvement in performance, despite the error bars due to the random nature of the smaller splits. Performance increases and is more stable as the split sizes increase as the variability of the tables is captured better on larger splits.

User-Facing Results In many ways, the problem of detecting a certain set of tables from a larger collection of pages is similar to a search-ranking problem, where we do not necessarily know that a given factor exists, and we need to show the likely candidates sorted in a logical order, without presenting extraneous pages. We use R-precision as the metric to determine how the table detection performance affects the results presented to the analyst (Beitzel, Jensen, and Frieder 2009). In summary, R-precision can be defined as the precision at the R^{th} position in the ranking of results that has R relevant documents. For example, if a factor table spans $R = 5$ pages in a filing, R-precision would be the number of correct pages returned in the first 5 results. This requires knowing exactly which factors correspond to

Filing	Average R-precision	Completeness
Filing 1	0.937	58 / 60 (0.967)
Filing 2	0.894	102 / 104 (0.981)
Filing 3	0.960	122 / 124 (0.984)

Table 1: User-facing results reflecting how Harvest performs as a *search engine*. Harvest found 282/288 tables (97.9 %) in these filings. Upon visual inspection, it was noticed that the tables missed were very small, some only 2 cells big with no borders, looking visually similar to a line of text.

which pages in a document. To this end, we collected logs from three rate filings that Harvest was used to process. We also measured *completeness* as the number of pages of factor rate tables that were detected vs the total number of factor rate tables that were in the filing. We provide the results in Table 1. We should also note that in some filings, there are semi-duplicate tables (for example, a table that shows the current factor, and a factor showing the changes in the factor table from a previous filing), and the user requires only one of them. That can result in extra pages being shown to the user, which affected the average R-precision in Filing 2. These values were deemed satisfactory by the end user (this is discussed further in the User Satisfaction section).

We also explored the effect of table detection performance (as measured by COCO metrics) on these user-facing metrics. The precision of the model used here was 0.95 and recall was 0.68. We then evaluated R-precision and completeness values at earlier stages of training to see the effect of different precision/recall values in R-precision and completeness. It was seen that the while the R-precision and completeness values can be somewhat stable for some filings, they drop significantly for others when the precision and recall values are a few percentage points lower earlier in the training. As such, to maintain the same level of satisfactory user performance, we found that it is necessary to maintain precision around 0.95, and recall around 0.68.

New Company Onboarding

A table detection model can benefit significantly from fine-tuning as noted previously. However, doing this for every company and potentially multiple business lines can be costly and add lead-time to adding a new company's tables in this system. From a scalability standpoint, it is important to minimize the effort required to train a new model on a new company's documents. This requires finding a way to efficiently adapt the existing models to each new company.

Domain Adaptation Domain adaptation is an unsupervised approach to adapting an existing model trained on a particular domain to a new domain (Long et al. 2015; Ganin and Lempitsky 2015; Gebru, Hoffman, and Fei-Fei 2017). The idea behind domain adaptation is to adapt an existing network to focus on domain-independent features for the task at hand via regularization that enforces constraints between the *source* (original) domain and the *target* (new) domain the model is expected to perform on. If the model can perform its task based on such domain-independent features, then it is expected to transfer better to new domains.

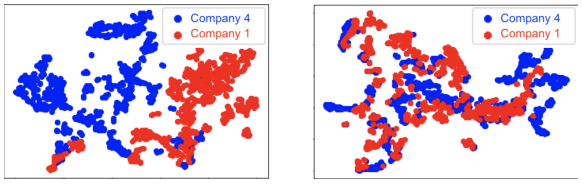

Figure 7: This t-SNE plot of features prior to the object classification layers of Faster-RCNN shows an example of domain adaptation on the features of two companies. The features in blue correspond to an image from a source domain, and those in red correspond to an image from the target domain. On the left are the features from the base model, which can be seen to differ significantly. On the right are domain-adapted features after training via labeled images from the source domain and unlabeled images from the target domain. These features are much more aligned.

For object detection, a similar approach was proposed by Chen et al. (2018). In this paper, it is noted that a model can fail to perform well on a new model due to image-level shifts such as illumination or style, or for object-level shifts such as changes in the appearance of the objects of interest in the image. To address this shift and adapt an existing model, they propose a training paradigm that adds some components to a Faster-RCNN model that penalizes the dissimilarity of features (at both image- and object-level) between the source domain (consisting of labeled images from the original training set) and the target domain (consisting of images from the new domain). The training is done in an adversarial fashion, and does not need labels for the target domain - if the features between the source domain and target domain are similar, then it is expected that the performance of the target domain be similar to that of the source domain. We use a number of labeled examples from the source domain and an equivalent number of *unlabeled* examples from the target domain to perform domain adaptation of the base models. Figure 7 shows how domain adaptation can align the features between the source and target domains.

In practice, we found domain adaptation to be very sensitive to parameter values, as it aims to balance the performance of the Faster-RCNN model on the labeled data, while trying to align the features between the labeled and unlabeled data. Finding a consistent point where these two competing constraints are balanced is a challenging task, and requires some experimentation. In general, we found that straightforward application of domain adaptation *without any labeled examples from the target domain* for training can slightly reduce the performance of the model on the target dataset. However, we observed that fine-tuning the domain-adapted model can exceed the performance of fine-tuning the base model. A two-step approach that involves unsupervised domain adaptation with supervised fine-tuning can allow the model to reach the fine-tuned performance of the base models with fewer labeled examples. Automating this pipeline can reduce the need for labeling and help onboard new companies faster.

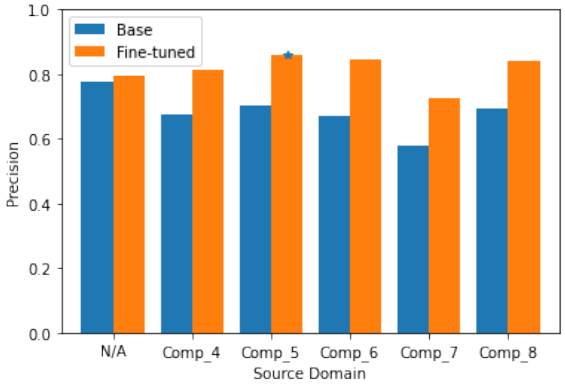

Figure 8: Precision values for domain adaptation with Company 1 as the target domain. 'N/A' indicates conventional fine-tuning without domain adaptation. Best results come from using Company 5 as the source domain.

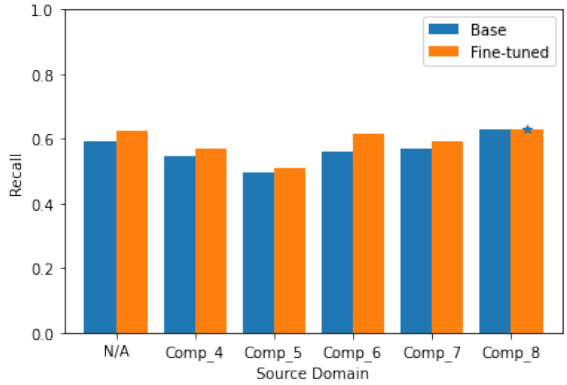

Figure 9: Recall values for domain adaptation with Company 1 as the target domain. 'N/A' indicates conventional fine-tuning without domain adaptation. In this case, highest recall is by using Company 8 as the source domain.

Another point that is worth mentioning is that domain adaptation performs best when the 'source' domain and 'target' domain during the adaptation process are similar. In other words, after the base model is trained, identifying the 'closest' domain to the target domain can lead to improved performance as the labeled examples from the source domain can be used for data augmentation during the fine-tuning stage; we can perform fine-tuning with labeled examples from the similar source domain as well as the target domain, further reducing the need for labeled examples. Figures 8, 9 show the effect of different source domains as co-training domains in a domain-adaptation setting for a target domain (Company 1). While domain adaptation can reduce the initial precision & recall values compared to the non-domain adapted model (See the 'Base' columns in the bar chart), jointly fine tuning this model afterwards with images from both the source & target domains can take advantage of the labeled source domain data and act as data augmentation, potentially leading to significant improvements, especially in precision. Similar conclusions were reached by Casado-García et al. (2020), namely that "fine-tuning works better when there is a close relation between the source and target task." As such, we are exploring ways to be able to identify good co-domains that would allow us to only use unlabeled data from the target domain for adaptation and estimating performance.

Table Extraction

Once detected, the information in the detected tables need to be extracted in a format that will allow analysis of this data. After some exploration, we decided to rely on the Amazon Textract service to extract the tabular data. Textract Table Detection was seen to be mostly reliable in extracting tabular information, with some caveats discussed below.

Textract

Amazon Textract is a cloud-based service that can perform OCR on a page and provide the results in a semi-structured fashion, including tables. In general, we found Textract to be a reliable service for Harvest use to extract the data in the detected tables. To see if we could rely solely on Textract for table detection, we used COCO metrics to compare the results of Textract table detection outputs and compare it to our results. While Textract precisions are comparable to the baseline results, out of the box it generally has better recall than the base model as seen in Figure 5. However, fine-tuning with as little as 50 images for a company can provide a significant (and necessary) boost to our precision and recall values, easily matching and sometimes significantly exceeding Textract's performance. This emphasizes the potential benefits of in-house models over one-size-fits-all approaches. As noted prior, we use the COCO metrics precision @ IoU > 0.5 and recall @ 10.

We also note that different companies can have significantly different precision and recall values, that seem to be challenging to both Textract and Faster-RCNN baseline models. This can be attributed to the widely diverse looks of the tables in each company, and the internal consistency of tables within a company's filings. This supports our conclusion that (1) Fine-tuning is essential to reach a high-level of performance, and (2) individual models for each company need to be employed to optimize the performance.

Optical Character Recognition OCR plays a significant role in the accuracy of data. We have spent considerable time on looking at ways to optimize OCR performance. As these filings are, in general, born digital, using 300 dpi when converting the documents to images seems to result in very few errors. Any errors from Textract are mostly due to issues with table structure recognition rather than the actual text. Furthermore, an analyst further validates the data before it is stored in a database to ensure the quality of the data. In the future, as the system is used to collect and store more tabular data, we plan to explore ways to automatically flag significant deviations from existing data that may be due to OCR issues.

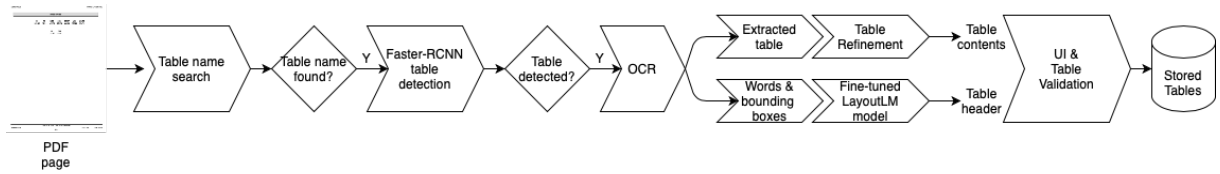

Figure 10: Harvest processing of a filing page

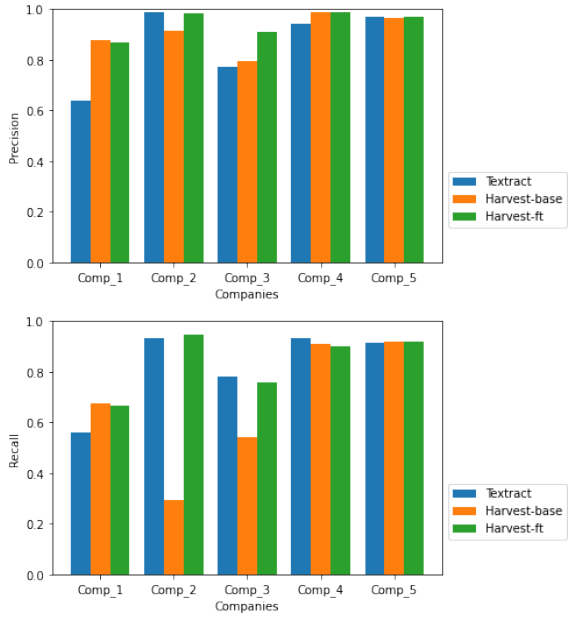

Figure 11: Precision (top) and Recall (bottom) for table detection performance of Textract vs Harvest.

Table Refinement We noted that for larger tables, especially those that spanned most of a page, Textract struggled to find accurate boundaries for the cells, occasionally merging whole rows or columns. To tackle this issue, we used a traditional computer-vision approach. This is performed by scanning lines through the image and marking areas where the lines do not intersect with the detected text lines to get a binary set of values for row/column boundary coordinates. This set of coordinates is smoothed with a Gaussian filter, followed by non-maximum suppression and peak detection to estimate cell boundaries. The contents are then reassigned to the new cells. We found that this can significantly improve table structure output for large and dense tables.

Table Metadata

In addition to extracting the tables, analysts also need to be able to determine the particular title of each table as the calculations of rates for particular factor can be split into multiple tables that contribute in different amounts. Furthermore, determining table names is used for combining multi-page tables; some factor tables can span over ten pages and need to be combined into a single table to query. To tackle this problem, we initially used the Faster-RCNN model trained

Figure 12: LayoutLM used to detect document structure. Red indicates 'Other' text, green indicates headers/titles and blue indicates the tabular data.

for tables in a multi-class fashion, using the extra labeled headers and captions. However, the performance for this approach was not satisfactory, which we believe to be due mainly to (1) faster-rcnn does not take into account the relationship between the various objects in the page, and (2) The captions and headers ended up just being text areas that look very similar to other areas of text in the page.

LayoutLM In order to make use of the page layout and the expected spatial relationships between these components of a table, we decided to use a LayoutLMv1 model (Xu et al. 2020). LayoutLM is a language model that incorporates two-dimensional positional embeddings of words to make use of spatial context, where different tokens may lie with respect to each other in a document layout. This is opposed to standard language models such as BERT (which LayoutLM is based on) that only take into account the position of words in a 1D word stream. Optionally, LayoutLM can also take advantage of image embeddings in addition, allowing it to capture richer context than regular language models.

As noted previously, tables can vary significantly between companies, necessitating separate models for optimal performance. However, it is possible to use the same labeled datasets that were used for training the table detection models. A LayoutLM-based sequence classifier model was trained by labeling each word in a document as belonging to the fields 'Other', 'Header', or 'DataTable', the classes we used in table detection. This was done by performing OCR to extract bounding boxes for words; and labeling each word with the label of the area labeled for table detection that its bounding box had the most overlap with. We show an exam-

Type	Precision	Recall	F1-Score
DataTable	0.65	0.69	0.67
Header	0.78	0.85	0.81

Table 2: LayoutLM for detecting table headers

ple of the LayoutLM predictions in Figure 12 and the precision/recall results in Table 2.

In general, the LayoutLM model was seen to work satisfactorily. With some post-processing (e.g., assign all words in a 'row' the same label as the majority, a heuristic that seems to reliably hold for filing documents), these results improve in practice. Note that for table detection Faster-RCNN outperforms LayoutLM.

Cloud Implementation

The process a particular page undergoes is shown in Figure 10. Harvest was deployed in Amazon Web Services as a cloud-based platform, allowing integration with other services and easier connectivity to other background tasks. Filing documents can span over a thousand pages and contain hundreds of pages of tables. Processing each page sequentially through all the processes of text extraction / table detection / table extraction / metadata detection can take up to a half hour or more for a large filing. To optimize the user experience, we have developed a parallel cloud-based process that allows a filing document to be processed in a massively parallel fashion. The process is summarized below:

1. Analyst uploads the filing PDF document to Harvest.
2. The PDF is split into pages, and each page is sent to a separate AWS Lambda that extracts the text and process to look for the factor names in that page.
3. The pages that were determined to have the names of the tables in the analyst-provided list of factor tables are queued for further processing.
4. AWS Lambda functions run table detection models on queued pages. Pages where tables were detected are then sent to AWS Textract for table extraction and refinement.
5. The pages with detected tables are sent to a further set of AWS Lambda functions that apply the LayoutLM model to extract table headers.
6. Results are combined and presented to the analyst via the UI once all the pages are processed.

The communication between processed is established via a combination of message queues and an in-memory key-value pair database. This approach cuts down the processing time to 2-3 minutes instead of 30-40 minutes for larger filings. The architecture is presented in Figure 13.

End User Experience

The goal of Harvest is to use various machine learning models to greatly simplify and streamline what is currently a time-consuming manual task. Its adaptability at a larger scale depends on the performance of the machine learning models as well as the satisfaction of the system's users. Harvest also includes a user interface that allows the user to view and to easily make edits to the extracted tables.

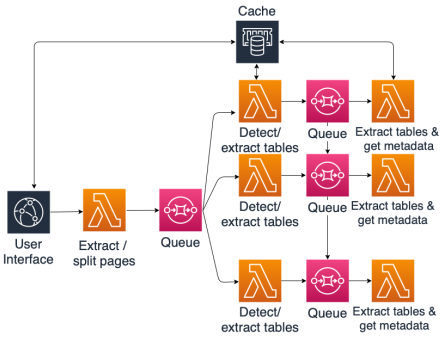

Figure 13: Harvest cloud deployment

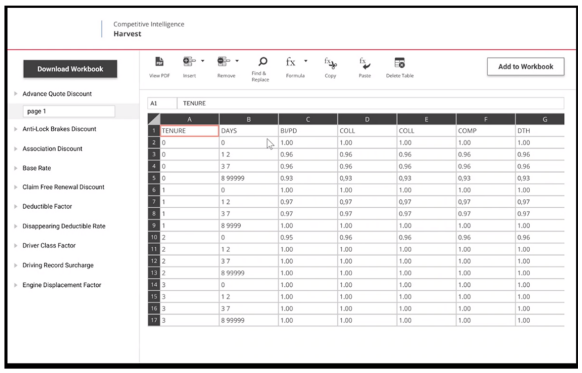

Figure 14: Harvest main user interface

User Interface

The Harvest user interface resembles an online spreadsheet program. The user starts by selecting the list of factor tables they are interested in and uploading a PDF. The document is processed within a couple of minutes, and the user is presented with the user interface that allows them to review the tables and compare them to the filing PDF, with links to the pages each table was detected in. The user can make edits or remove tables, and when satisfied commit their changes which sends the results to a cloud-based database. A screenshot of the user interface is shown in Figure 14, displaying a factor rate table extracted from a filing and the editing interface that the analyst can use to correct any errors.

User Satisfaction

Currently, there is one analyst using Harvest. We evaluated their satisfaction with Harvest using a modified System Usability Scale (Brooke et al. 1996) that added 8 more questions regarding the usability of Harvest (pertaining to 4 dimensions, bidirectionally coded). The SUS score was 77 (a score over 75 is usually considered *good*), and the user noted they were 'Very Likely' to recommend Harvest to others.

Various business units and operating companies within our enterprise have expressed significant interest in using Harvest for their own similar processes, and we expect the user base to significantly increase. We can foresee the system used by dozens of people for future deployments of the application in the enterprise; this was one of the reasons we

focused on scalability. Further, Harvest can reduce the field-expertise currently necessary for the manual task as the verification can be done with minimal training, saving the analysts' time to focus on data analysis.

Conclusions

Even as more data is born digital and stored in a structured fashion, we envision that many conventional industries such as insurance and financial industries, will need to handle unstructured data, whether it is coming from other companies or their customers. To our knowledge, Harvest is the first scalable workflow that can allow widespread extraction of tabular data from filing documents in a cost-effective and supervised fashion. While our initial focus was based on an existing business needs, similar architectures can be incorporated into other workflows or (common) business scenarios where there are large amounts of unstructured data that need to be *reliably* captured with minimal effort.

Future Directions

While Harvest can significantly reduce the time it takes to process filing documents, it still relies on a human to validate the results, as the accuracy and completeness of the tables is of crucial importance for the business. We will be exploring ways to further automate this validation process by detecting entries that may not fit in the expected values. The data collected through Harvest-processed filings can be used to identify trends in field-values, flag unusual changes to analysts and further improve the efficiency of this process.

We are also interested in improving the on-boarding process for new companies or business lines by exploring methods that can help better identify matching source domains to further boost the zero- or few-shot table detection performance. In addition, we plan to explore large-scale pre-training of the LayoutLM models we used to identify table metadata to improve detection of table headers and captions. This model that is pre-trained on filings can also be used to robustly identify filing source, date and state from PDF files as well, further moving towards our goal of a fully-automated system that can identify potential errors and only alert the users to these issues.

References

Beitzel, S. M.; Jensen, E. C.; and Frieder, O. 2009. *Average R-Precision*, 195–195. Boston, MA: Springer US. ISBN 978-0-387-39940-9.

Brooke, J.; et al. 1996. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194): 4–7.

Casado-García, Á.; Domínguez, C.; Heras, J.; Mata, E.; and Pascual, V. 2020. The benefits of close-domain fine-tuning for table detection in document images. In *International Workshop on Document Analysis Systems*, 199–215. Springer.

Cesarini, F.; Marinai, S.; Sarti, L.; and Soda, G. 2002. Trainable table location in document images. In *Object Recognition Supported by User Interaction for Service Robots*, volume 3, 236–240. IEEE.

Chen, Y.; Li, W.; Sakaridis, C.; Dai, D.; and Van Gool, L. 2018. Domain adaptive faster r-cnn for object detection in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3339–3348.

Ganin, Y.; and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, 1180–1189. PMLR.

Gao, L.; Huang, Y.; Déjean, H.; Meunier, J.-L.; Yan, Q.; Fang, Y.; Kleber, F.; and Lang, E. 2019. ICDAR 2019 Competition on Table Detection and Recognition (cTDaR). In *International Conference on Document Analysis and Recognition (ICDAR)*, 1510–1515.

Gebru, T.; Hoffman, J.; and Fei-Fei, L. 2017. Fine-grained recognition in the wild: A multi-task domain adaptation approach. In *Proceedings of the IEEE International Conference on Computer Vision*, 1349–1358.

Kasar, T.; Barlas, P.; Adam, S.; Chatelain, C.; and Paquet, T. 2013. Learning to detect tables in scanned document images using line information. In *12th International Conference on Document Analysis and Recognition*, 1185–1189. IEEE.

Kavasidis, I.; Pino, C.; Palazzo, S.; Rundo, F.; Giordano, D.; Messina, P.; and Spampinato, C. 2019. *A Saliency-Based Convolutional Neural Network for Table and Chart Detection in Digitized Documents*, 292–302.

Li, M.; Cui, L.; Huang, S.; Wei, F.; Zhou, M.; and Li, Z. 2019. TableBank: A Benchmark Dataset for Table Detection and Recognition. arXiv:1903.01949.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, 740–755. Springer.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. 2015. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, 97–105. PMLR.

Paliwal, S. S.; Vishwanath, D.; Rahul, R.; Sharma, M.; and Vig, L. 2019. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, 128–133. IEEE.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Schreiber, S.; Agne, S.; Wolf, I.; Dengel, A.; and Ahmed, S. 2017. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, 1162–1167. IEEE.

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*.