Custom Entity Extraction: An End-to-End Machine Learning Pipeline Tailored to Unique Customer Data

CS4991 Capstone Report, 2023

Madelyn Khoury
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
mgk5ybb@virginia.edu

ABSTRACT

Appian Corporation, a producer of enterprise software, hoped to improve the performance of its intelligent document processing (IDP) capabilities, particularly for documents that are uniquely formatted, such as documents used by their high-volume customers. To improve accuracy of the entities extracted from customer documents, my team and I developed a custom entity extraction (EE) feature, allowing clients to use their own model rather than a model shared among all Appian customers. I adapted and streamlined the BROS model—an open-source EE model capable of better predictions—for use by customers. Additionally, I built a Kedro machine learning pipeline to integrate this new model into the overall document-processing workflow. My team and I created an end-to-end inference workflow using the custom EE model. Future work required to complete the custom EE feature includes implementing a training workflow for the model.

1. INTRODUCTION

Imagine a large company that processes thousands of invoices per month. This company hopes to automatically extract information like the date and purchase amount (also known as "entities") from each invoice so that employees don't have to spend dozens of hours doing so manually. The company uses Appian's IDP tools to process the invoices—only to realize that the date extracted from many of the invoices was incorrect.

This example reveals the impacts of inaccurate IDP, specifically inaccurate entity extraction. When entity extraction fails, a human must go through *all* the documents to verify correctness of information and make changes if necessary. In the previous example, one would need to verify/edit the date field extracted from each invoice. Doing so defeats the purpose of automated document processing and prevents customers from being able to automate a core business process, which means that Appian cannot fully deliver its promise of business automation. This outcome results in frustrated customers, negative economic impacts for Appian, and wasted time. In some cases, errors might not be caught – either because the customer has not realized that the model performs poorly or because human verifiers make mistakes. Such errors could lead to inaccurate data being saved, putting the customer's business processes at risk. To reduce the number of errors during entity extraction, my team and I began implementing a feature to allow customers to train a custom model.

2. RELATED WORKS

Companies that produce software products for their customers (called "Software as a Service", or SaaS, companies)

have long considered the tradeoff between training a single, general purpose model and training a unique model for each customer [5]. General models may make use of more training data, finding overall patterns among all customers' data and performing well as a result [7]. However, training with inaccurate data can decrease model performance; so, when one customer's data differs from the data used to train the general model, a specialized model may be the best option [5].

There are many models to choose from; for this case, a language model, which learns how words are strung together to create sentences and thus to understand documents, is relevant. Particularly, the Bidirectional Encoder Representations from Transformers (BERT) model achieves good performance because it encodes an understanding of the context entirely surrounding a word, not just an understanding of what comes before or after the word [4].

The BERT Relying On Spatiality (BROS) model builds upon BERT to address the issue of entity extraction. Specifically, BROS encodes an understanding of document layout, storing the relative location of one word to every other word. BROS requires fewer parameters than other entity extraction models and achieves better performance on key information extraction tasks [6].

The training and use of a machine learning model consists of multiple steps. Frameworks such as Kedro are intended to help separate steps into discrete "nodes" in a pipeline [1]. The Kedro framework has been employed in industrial software contexts in the past. For example, scientists at NASA's Ames research center chose Kedro as the basis for a machine learning pipeline because it allows for "abstract data access" [2]. Amblard, et al. used Kedro to create a pipeline for model training [2]; in contrast, this paper discusses the creation of a pipeline for model inference.

3. PROJECT DESIGN

To implement the ability to perform inference with a custom EE model, I created a Kedro pipeline capable of making predictions with a specified custom model based on the BROS model. Then, I integrated this pipeline into the existing document extraction workflow.

3.1 Document Extraction Workflow

Appian's existing inference workflow for document extraction consists of several steps. First, however, it will be useful to discuss how the workflow is instigated. Figure 1 shows a flow chart representing the typical use case for document extraction [3]; this chart represents a customer business process in which data is extracted from documents, manually verified, and then written to a database. When the "Extract from Document" node is reached, a request is sent from the Appian platform that users interact with, called Appian Enterprise (AE). This request is sent to the machine learning admin service (MLAS), which is responsible for starting workflows for different machine learning capabilities. Document extraction inference is just one of these workflows.

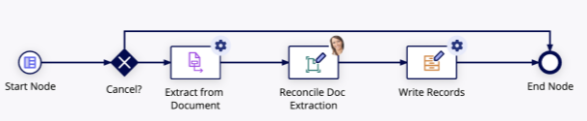

Figure 1. Typical Document Extraction Process Model

The current document extraction workflow begins with creating a Kubernetes pod in which to run an optical character recognition (OCR) model. The model is run, producing "text extractions," or the text contents of a PDF previously unreadable by computers. Next, the text extractions are passed to Appian's general EE model, along with the desired information to extract. Another Kubernetes pod is created, and the general EE model performs inference; identifying a piece of text (i.e., "extracted

entity") from the document to match each desired piece of information. Finally, the extracted entities are passed back to AE, where they can be used in later steps in the process. The steps of the workflow are shown in Figure 2.

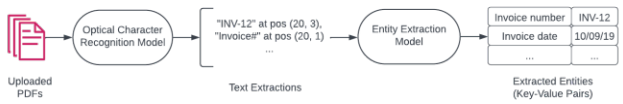

Figure 2. Document Extraction Workflow

3.2 Modifications to Workflow

Implementing inference with a custom EE model necessitated replacement of the general EE model with a custom EE model. Thus, most of the workflow could be preserved, but it was necessary to create a program to run inference with the new model and to deploy that program with Kubernetes. This program took the form of a data pipeline, implemented with the Kedro framework. In addition, I created functions in AE that would read the results of EE with the new model, since the results were in a slightly different form than the EE results from the general model.

3.3 Kedro Pipeline for Inference

The process of running inference with a machine learning model involves several steps. My team elected to use the Kedro framework to create a data pipeline, where each step of the process could be executed in a separate node. First, the text extractions from the OCR model had to be loaded in, cleaned, and converted to the required format for use in the custom EE model. Second, I created a dataset object which would package the data into sections and make further changes to the data format so it could be used by the custom model that we chose, the BROS model. Next, the model itself had to be instantiated. As each customer will already have a trained model, I loaded in the trained model's weights from a binary file and created an object to represent the custom EE model, which could then be used to make predictions. Finally, I performed inference using the model before finishing the process by converting the BROS model's predictions into the format we wanted them to be in. Each of these steps became a node in my Kedro pipeline (see Figure3).

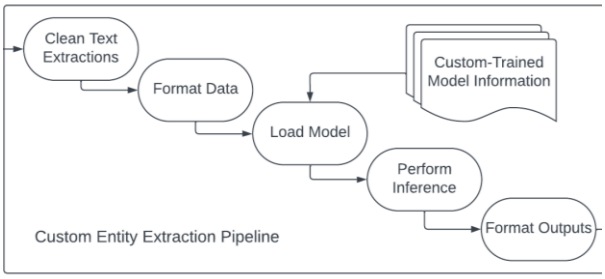

Figure 3. Kedro Pipeline

3.4 BROS Model Refactoring

The final design choice I made to complete the inference workflow was to refactor and streamline the BROS model. The BROS model was produced as part of a research project by Clova AI, an open-source AI organization. The BROS model code by itself was not production-ready for several reasons. First, many functions were long and hard to understand; I addressed this by making variable names more descriptive and decomposing large functions into smaller ones that each completed only a single operation. Second, the BROS model was first created to be used with data in multiple different formats. However, Appian had decided on a standard data format, rendering most of those formats unnecessary. I removed all code pertaining to data formats that Appian was not using. Additionally, I removed other redundant/unused variables.

4. ANTICIPATED RESULTS

There are several anticipated benefits of the entire custom EE feature. First, the BROS model performed better than Appian's current general model in preliminary tests, achieving a 27% increase in accuracy, as measured by the model's F1 score. We anticipate that custom BROS models trained and used as

part of the custom EE feature will have a similar performance increase. This increase will be particularly beneficial for Appian's customers with uniquely formatted documents, since they have, to date, received the worst performance with the general EE model. These customers tend to be the ones that have the most documents to process, so the custom EE feature is expected to greatly reduce the number of manhours required to process documents.

Additionally, Appian currently performs about 6,000 extractions per month for over 170 customers, all of whom will be positively impacted by the option of training a custom EE model. I anticipate greater adoption of IDP in general—manifesting as an increase in both IDP customers and number of extractions per month—since several customers currently do not utilize IDP due to its poor performance.

5. CONCLUSION

In this project, I implemented, integrated, and tested a machine learning pipeline that allows customers of Appian Corporation to process documents using a model trained on their unique data. Further, I adapted a state-of-the-art EE model, BROS, to be production ready for use in this pipeline.

As custom EE is anticipated to isolate information from documents more accurately, more customers will be able to adopt Appian's IDP functionality and to invest less human labor into verifying document processing results. This will allow Appian to provide true automation of document processing, in turn causing Appian to gain more customers, achieve better customer satisfaction, and see increased profits.

6. FUTURE WORK

Though the inference workflow for custom EE has been completed, it is untested in a production environment. For the custom EE feature to be released to customers, an additional workflow for training models will have to be developed, as customers will have to train their custom model before using it. Then, further tests will have to be conducted to verify that the two workflows interact well with one another.

7. ACKNOWLEDGMENTS

Thank you to Steph Colen from Appian Corporation for overseeing my technical project.

REFERENCES

[1] Sajid Alam, Nok Lam Chan, Laura Couto, Yetunde Dada, Ivan Danov, Deepyaman Datta, Tynan DeBold, Jitendra Gudaniya, Jannic Holzer, Stephanie Kaiser, Rashida Kanchwala, Ankita Katiyar, Ravi Kumar Pilla, Amanda Koh, Andrew Mackay, Ahdra Merali, Antony Milne, Huong Nguyen, Vladimir Nikolic, Nero Okwa, Juan Luis Cano Rodriguez, Joel Schwarzmann, Dmitry Sorokin, Jo Stichbury, and Merel Theisen. 2023. Kedro. Retrieved from https://github.com/kedro-org/kedro

[2] Alexandre Amblard, Sarah Youlton, and William J. Coupe. Real-Time Unimpeded Taxi Out Machine Learning Service. In *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2021-2401

[3] Appian. 2023. Build a Doc Extraction Process with AI Skill. Retrieved from https://docs.appian.com/suite/help/23.3/doc-extraction-tutorial.html

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805

[5] Yonatan Hadar. 2021. Should I train a model for each customer or use one

model for all of my customers? *Medium*. Retrieved October 14, 2023 from https://towardsdatascience.com/should-i-train-a-model-for-each-customer-or-use-one-model-for-all-of-my-customers-f9e8734d991

[6] Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. BROS: A Pre-trained Language Model Focusing on Text and Layout for Better Key Information Extraction from Documents. https://doi.org/10.48550/arXiv.2108.04539

[7] Samuele Mazzanti. 2023. What Is Better: One General Model or Many Specialized Models? *Medium*. Retrieved October 14, 2023 from https://towardsdatascience.com/what-is-better-one-general-model-or-many-specialized-models-9500d9f8751d