A machine learning pipeline for document extraction

Chin Hang Lun[1*], Thomas Hewitt[1] and Song Hou[1] discuss how machine learning can be used most effectively to classify documents.

Introduction

Each year the geoscience industry creates huge volumes of documents containing data, research, and plans, which describe valuable subsurface assets. These documents contain a wealth of knowledge in the form of text, figures, and tables which are intended to be read by humans and cannot be easily queried or extracted. It is important to incorporate all sources of data to create a holistic model of the subsurface and therefore reduce exploration and development risks. Key to the successful extraction and transformation of data is an understanding of the nature of the data that exists within a corpus of files. This is achieved by labelling files based on the data that they contain so that they can be grouped and prioritised for extraction. Some easy wins for labelling can be achieved in the case of well organised datasets where key information about the file is contained in its file path. However, this information is often not consistently captured and hence requires many search terms to group files. In less organised datasets it is often the case that key information is not captured within the path or is of insufficient detail to provide a useful label. In these cases, the only solution to reliably classify files is to manually open and review each document in turn. This is feasible for a small number of files but as dataset sizes grow, the time required to understand what is contained in a dataset increases dramatically.

For large datasets, it would be more efficient to develop an automated way to find the key terms within a piece of text, identify the figures and tables on a page, and classify the document, in order to help domain experts explore the vast document landscape.

In recent years, there have been huge advances in machine learning, computer vision, and natural language processing (NLP). These disciplines are concerned with the development of algorithms that enable machines to understand images and human-written languages. In this article, we discuss how machine learning is used at CGG to classify documents. Machine learning models are just one component of a larger pipeline which includes other data classification, data extraction, and data curation technologies. The pipeline can be described by a directed acyclic graph (DAG). See the example in Figure 1, which sets out the components or steps in the pipeline as well as the dependencies between each component. A dependency arises when a step requires the output of other steps. For example, the text on a page of a document needs to be extracted (using optical character recognition (OCR), for example) before an NLP model can be applied, and the figures need to be located before they can be classified. The automated pipeline is run on a user-specified dataset, which can consist of various file types, such as PDF, Word doc, Excel and CSV, and different image files, such as TIFF and PNG, etc. After initial ingestion, each of the components will be triggered in turn automatically as soon as its dependencies have completed successfully.

There are two advantages to our automated approach. Metadata in the file path or manual labelling can only support a single high-level label. Using the pipeline, a much more granular classification of the contents can be achieved at a paragraph or page object level. Secondly, many documents can be processed much more quickly. As an illustration, a previous project took 16 people 14 months to complete, whereas a recent equivalent project using the pipeline took 2 people 1 month, including computing time. This is a hundred times faster!

This article will focus on the machine learning steps in our pipeline, namely *document layout analysis*, *image classification*, *named entity recognition* and *table cell classification*. In recent times, machine learning has become increasingly dominated by deep learning models. These are data-hungry algorithms that need to be trained on large volumes of labelled input data. For

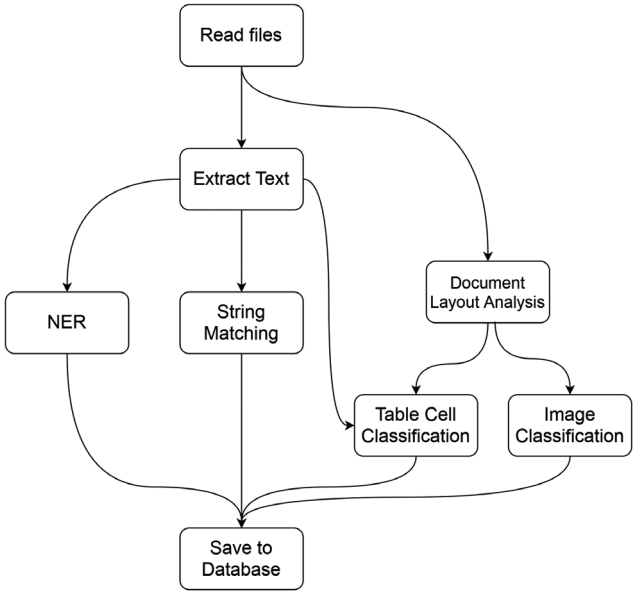

Figure 1 An example of a DAG describing the classification pipeline. The workflow starts from the top and an arrow indicates the direction of the flow of data.

[1] CGG
[*] Corresponding author, E-mail: chinhang.lun@cgg.com
DOI: 10.3997/1365-2397.fb2022016

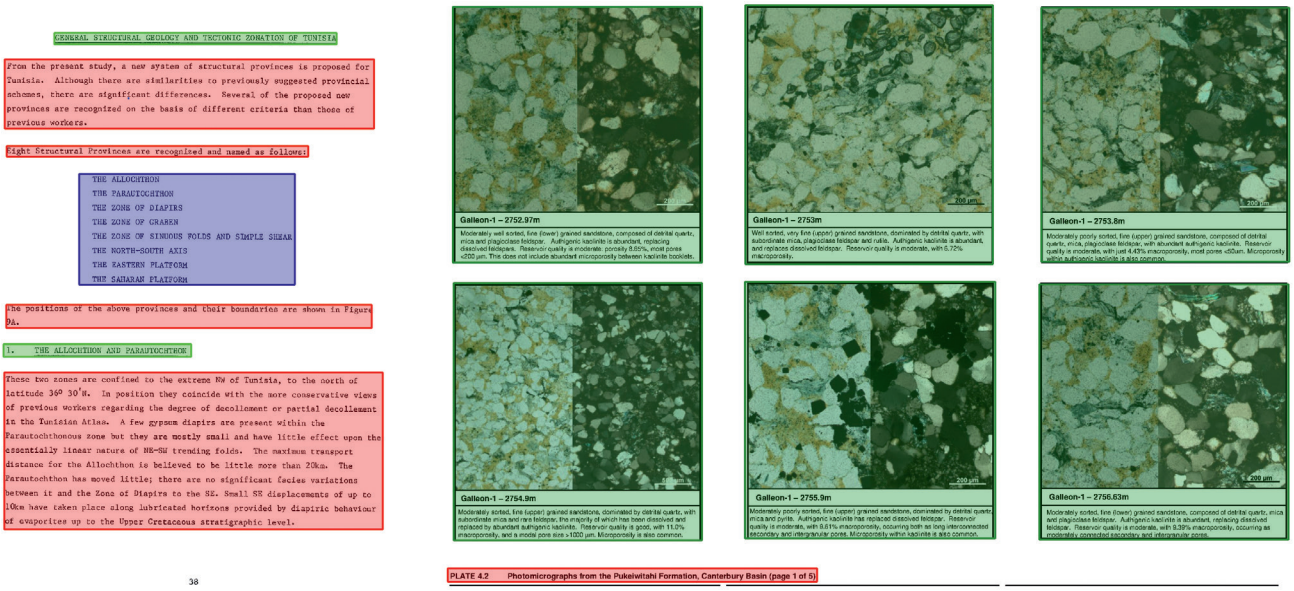

Figure 2 Example predictions of object detection. The left-hand image shows the detection of paragraphs (red), lists (blue) and titles (light green). The right-hand image shows the detection of figures (dark green).

this reason, part of the discussion will focus on how we obtain high-quality training data to train our models. Finally, we discuss how we deploy the machine learning models and the pipeline to process documents at scale.

Document layout analysis

Documents in geoscience and geology are often complex, containing not only text but also figures and tables which often reference each other. A first step towards machine document understanding is the ability to identify the components, such as paragraphs of text, tables, figures, title, and the table of contents on each page of the document – a task called document layout analysis. The identified components can then be further processed by the appropriate algorithms, such as image classification and table cell classification for the figures and tables, respectively, which we will discuss in subsequent sections.

The document layout analysis task is performed with a computer vision algorithm. Computer vision is a scientific field that deals with computer understanding of images or a sequence of images. The goal is often to automate tasks, such as image recognition, instance segmentation, and object detection. This identification may come naturally to humans in most scenarios, but it can be difficult or even impossible to describe how they are accomplished. The task we will focus on is object detection which involves predicting the bounding box of an object in an image and classifying the object within the bounding box. In the context of document layout analysis, the image would be of a document page and the classes of objects would be paragraphs, tables, and figures, etc.

Since 2012, after a convolutional neural network (CNN) called AlexNet (Krizhevsky et al., 2012) won the ImageNet challenge, CNNs have been the go-to model in computer vision. In our workflow, a model called Faster R-CNN (Ren et al., 2015) is used. The model treats the task as a supervised machine learning problem, which means that the model is trained using images, together with the desired predictions for each training image.

There are many publicly available datasets for document layout analysis; the largest ones being PubLayNet (Zhong et al., 2019) and DocBank (Li et al., 2020). However, these datasets were drawn from a narrow range of sources: the former from medical papers in PubMed and the latter from physics, mathematics, and computer science papers from arXiv. Moreover, they are all LaTeX-generated and relatively new, which means that examples from these datasets can look very different from the reports and papers in geology, which could be many decades old. It is therefore not surprising that the model pretrained on these datasets does not perform very well on geological documents. For this reason, we have created a new dataset by sampling a subset of document pages from our internal corpus and having them annotated by CGG domain experts.

The number of examples in our training set is insufficient to train a deep learning model from scratch, but we can fine-tune the model that was first trained on DocBank (or PubLayNet). One way of improving the performance of our model is to use more high-quality training data. However, as manually labelling a dataset is time-consuming, we have developed a method of generating synthetic document pages to complement our manually labelled dataset.

Training the model on our own dataset and then on the synthetic data greatly improves the performance compared to a model trained purely on the public dataset. Examples of predictions made by our model are shown in Figure 2.

Image classification

Once the location of the figures has been identified, the figures are passed on to a computer vision model for classification. A figure in a document can be a multitude of things, such as a graph, a seismic image, a core photo, or a thin section image, and each contains different types of information that would require a different process to extract useful data. Image classification is therefore a necessary step to allow the pipeline to automatically trigger the appropriate workflow to further process the image.

CNNs are the state of the art in image classification. An example is EfficientNet (Tan and Le, 2019). This is trained on labelled figures from CGG's database of figures. Figures in geoscience documents are often complex and a composite of multiple smaller figures of different types. We found it beneficial to first further locate the individual subfigures within a larger collection (see Figure 3) using an object detection model like the one used for document layout analysis before cropping the subfigures and feeding them into an image classification model.

With the different types of figures classified, the pipeline can then apply the appropriate algorithm to process the figure. In the case of core photos CGG has developed a machine learning model to predict the lithology of the core for each pixel in the image. This is performed by first segmenting the core from the background and then using the RGB values of each pixel of the isolated core to predict the lithology. This algorithm for processing core images can be applied independently of the pipeline to raw collections of core images. In such a case, we take a repository of core images at different depths, segment the core from each image, read the depth information from either the filename or by performing OCR on the core image, optionally stitch the core images together, and then run the lithology prediction. Figure 4 shows a diagram of such a workflow.

Natural language processing

Natural language processing (NLP) is a set of methods that enable machines to process human-written languages. Examples of tasks include text generation, text summarisation, text classification, and named entity recognition (NER), the latter of which will be the focus of this article. NER is a task concerned with identifying spans of text which constitute a named entity. A named entity or entity is anything that can be referred to with a proper name, such as a person or location, but can also include numerical values, such as dates and measurements. A typical application in geology is to identify terms such as well names, formation names, biostratigraphic taxa, or certain scientific measurements.

State-of-the-art NLP models are now dominated by deep learning architectures, particularly transformer transformer-based models (Vaswani et al., 2017), such as BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020), which contain billions of parameters. They are typically first pretrained for the task of language modelling using billions of texts before being fine-tuned for a downstream task using typically a much smaller amount of data. Language modelling involves predicting the next word in a sentence given the previous word or sequence of words or, in the case of mask language modelling, predicting the missing words given its surrounding context. Training a model to do this from

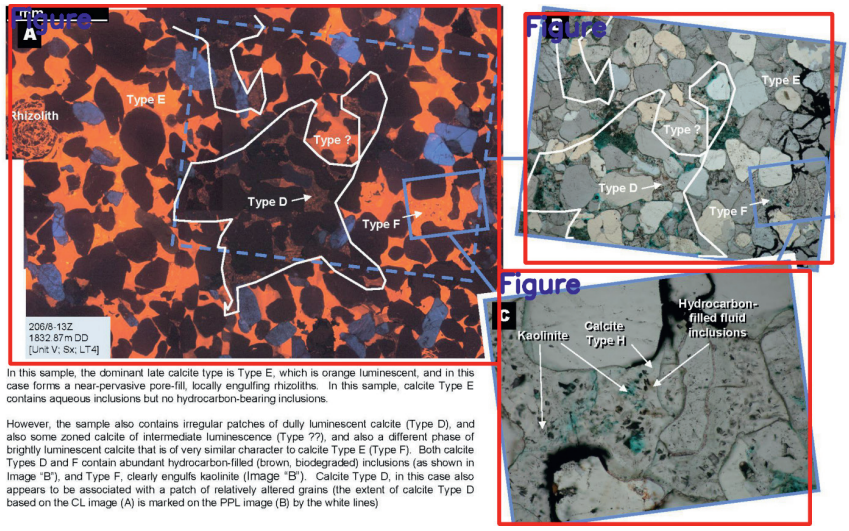

Figure 3 Locating the constituent figures inside a larger figure. Each of the subfigures are then cropped and classified.

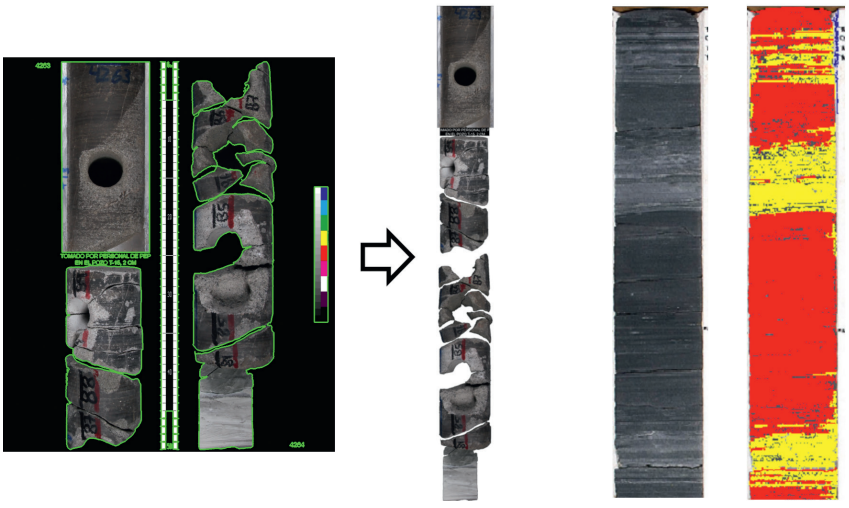

Figure 4 Left: The core is segmented from the background and stitched together. Right: A core sample and its predicted lithology; different lithologies are represented by different colours.

scratch requires a huge volume of text, but the good news is that there is no shortage of written text and no manual labelling is required because words in any sentence can be masked out.

Remarkably, experiments have shown that a single pretrained model can be fine-tuned to achieve a state-of-the-art performance on a wide variety of NLP tasks. One reason why these models are so effective is that by training a neural network for language modelling, it can learn to produce meaningful representations of words. To understand what this means, recall that for neural networks to be able to work with text, its constituent words need to be converted into numbers. Critical to the success of any NLP model is an encoding of words into numbers that are semantically meaningful. To illustrate this, consider a one-hot encoding of words in a fixed vocabulary of N words where each word is represented by an N-dimensional vector of zeroes with the exception of a single 1 in the ith position for the ith word. In this case, all the words will have a vector representation which are all equally spaced apart and therefore cannot capture any semantic meaning. Methods of generating word embeddings, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), solve this by representing words as dense vectors in a lower dimension (compared with the size of the vocabulary) by using a supervised learning technique involving neural networks. The vectors produced by these techniques seem to be able to capture the semantic relationships within a language; it is possible to experiment with publicly available word embeddings and check the following and other similar analogies: let K, M, and W be the vectors for the word 'king', 'man' and 'woman' respectively then the result of $K - M + W$ is approximately the vector for the word 'queen'.

However, with word embeddings, each word is considered independent and will have the same vector representation, irrespective of the context. For example, the word 'bank' can mean a riverbank or a bank in which one deposits money but with Word2Vec and GloVe the word would have the same vector in both contexts. Transformer-based language models consider the entire sentence to map each word (tokens to be precise) in a sentence into a vector. The vector for each individual word will vary depending on its surrounding words. We say that the model produces contextual embeddings of the words in the input sentence.

Often in geology and geoscience, the same word can have a very different meaning compared to its usage in everyday English. Good examples are the words 'play' and 'well'. As a result, we have found it beneficial to further train a model for the language modelling task on our geological corpus so that the model can see the various usages of the more specialised scientific terms. Pretraining on the geological corpus has increased the performance of the model on the downstream task of NER.

We treat NER as a supervised machine learning task which means that we need a labelled training dataset. As manually labelling many texts is time-consuming, we have developed an automated approach using as input our large geological corpus and our comprehensive taxonomy of terms that has been refined over the years by our domain experts. After the text has been extracted from the raw files, such as PDFs, string matching is performed on the extracted text to find occurrences of the taxonomy terms. We also complement this with rule-based pattern matching to tag anything not in the taxonomy, for example, using regular expressions to tag any well names which take the form NAME-NUMBER, such as 'Well-1'. While string matching and pattern matching manage to find entities in the text, this is not exhaustive because there may be spelling mistakes in the taxonomy or errors in the OCR which prevent entities from being matched and owing to many variations in naming conventions, it is difficult to implement all possible patterns for matching. More importantly, a taxonomy and any rules-based system are static and cannot adapt automatically to new names and new naming conventions. All of this necessitates the use of NLP techniques which consider the meaning of the input text to recognise entities.

With the dataset created we fine-tune our geological corpus-pretrained model to perform NER. Example predictions are shown in Figure 5. The model manages to identify entities that

Some Geli Khana beds show an abundance of the macrofossils Claraia sp. GENUSSPECIES , Costatoria sp GENUSSPECIES . and ? Bakevellia sp. GENUSSPECIES_NEW (found at K5 - 133), indicative of a typical Induan assemblage. The medium to well rounded, moderately sorted sandstone clasts at K5 - 134 could have the same origin as those described in dolomite filled vugs occur at K5 - 107. The topmost part of this calcareous section features a medium to dark grey bioclastic packstone, containing a characteristic Lower Triassic Tethyan fauna, including Costatoria ? costata GENUSSPECIES_NEW and Claraia sp. GENUSSPECIES (K5 - 109).

Examples of breached traps include UK 29/10a-2 WELL_NEW , 22/30a-1 WELL and 22/14b-3 WELL (Gaarenstroom et al., 1993). The hydrocarbons are likely to have migrated into the Chalk in these areas. Hydrocarbon shows throughout long intervals of the Chalk section in well 21/20a-1 WELL (dry hole) are consistent with vertical migration above this salt.

Figure 5 Example predictions on the test set. Labels with the suffix 'NEW' indicate that the entity is neither in the taxonomy nor found by pattern matching.

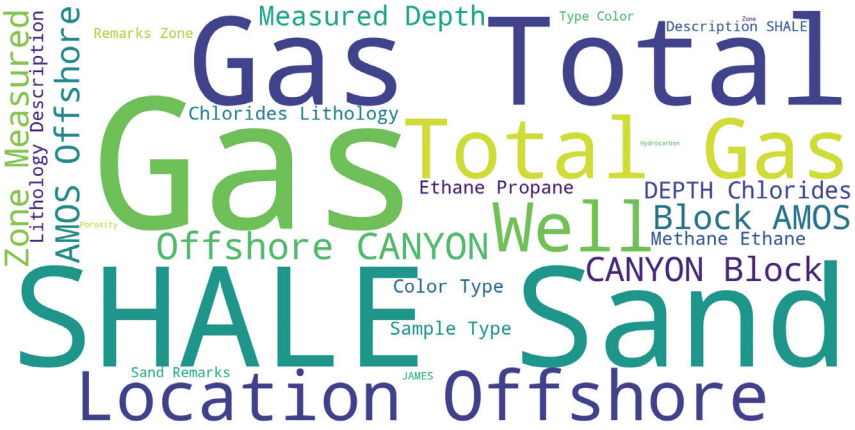

Figure 6 Word cloud of the key terms found in one document by string matching against our taxonomy and by NER.

Enron Entity Contact List for RFI's	nan	nan	nan	nan
(As of 4/29/02)	nan	nan	nan	nan
nan	nan	nan	nan	nan
nan	nan	nan	nan	nan
nan	nan	nan	nan	nan
Enron Entity	Commodity	Information Type	Primary Context	cc:
EPMI	Physical Power	AR/AP (Settlements)	Rebecca Grace	Juan Camarillo/Zakiyyah McClure
nan	Physical Power	MtM Valuation	Laurel Bolt/Ashley Fay	Sam Leuschen
nan	nan	nan	nan	nan
nan	nan	nan	nan	nan
ENA	Physical Gas	AR/AP (Settlements)	Margaret Dhont	A.K. Matheson
nan	Physical Gas	MtM Valuation	Kulvinder Fowler/Patrick Ryder	Greg Couch/A.K. Matheson
nan	nan	nan	nan	nan

Figure 7 Example predictions of the table cell classification task. Green – Data, Red – Header, Teal – GroupHead, Orange – Title, Blue – Other.

were not originally in the taxonomy and therefore provide a way for us to enrich our taxonomy.

Figure 6 shows an example of the key terms found in a document by string matching and by NER. These key terms can then be used to classify documents. Moreover, since we keep track of the location of where each term is found, we can achieve a more granular labelling of the document by classifying each paragraph individually.

Table cell classification

Tables and spreadsheets contain a wealth of data that are already in a structured but variable format. Transforming the contents of a table into useful data not only requires extracting the content of each individual cell, it also requires identifying the relationship between these cells. A first step is to identify the role a cell plays in the table. By knowing the header a data cell belongs to, it is possible to give meaning to the data and to infer facts. In the case of spreadsheets, as well as identifying cells in the table, we need to identify cells around it, such as notes and metadata, which may provide useful context to the table. It is also crucial to identify the headers in a table so that the table data can be merged with downstream processes.

We have explored two different approaches: one involving feature engineering and using traditional machine learning techniques, and the other using natural language processing.

In the first approach, each cell is associated with a set of features which are fed into a model to classify the cell into one of the cell types. Features are computed from the contents of the spreadsheets. These can be categorised into three types: first, features that are derived from just the cell itself, such as the number of characters in the cell, the percentage of the characters that are letters, digits, punctuations, or whitespace, whether the cell is purely numeric, whether the cell is empty as well as the row and column index of the cell. The second and third types of features are averages of the above features over cells in the same row and column respectively. A similar approach has been experimented in (Koci et al., 2019). In their work, the authors used features specific to Excel spreadsheets, such as whether the cell contains a formula. We decided not to use Excel-specific features since we want to keep our method sufficiently general so that it will also work for tables on a document page.

In the NLP approach, we treat the problem as a token classification task. More precisely, the contents in each of the cells are tokenised and all the tokens in each cell are fed into a deep neural network where the output is a classification of each token. The final classification of the cell is decided by a majority vote of its constituent tokens. The tokens are ordered row-wise, starting from the top-left most cell then proceeding along the row until the right-hand boundary of the spreadsheet and then continuing from the left-most cell on the row below. For the neural network, we used a language model called LayoutLM (Xu et al., 2020). This approach allows us to feed the row and column index of each cell into the model, thus incorporating the 2D structure of the table in the prediction, which would otherwise be ignored by the contents being treated as just a sequence of tokens. Example predictions are shown in Figure 7.

Deployment

As mentioned in the introduction, vast volumes of documents have been produced over the years in the industry that will need to go through the machine learning pipeline. The success of any machine learning solution hinges on the ability to deploy the models at scale. Another important requirement is being able to automate the management of computing resources. By this, we mean the ability to schedule machines to run certain steps in the pipeline and automatically scale up/down machine learning models; we do not want engineers constantly monitoring usages and manually starting applications. In addition, details of those resources should be concealed from the pipeline end-users; they should not need to worry about allocating computing resources.

For deployment, we use Kubernetes, an orchestration system that automates the deployment, scaling, and management of containerised applications. To orchestrate the workflow and create the DAG we use Kubeflow which is built on top of Kubernetes. In Kubeflow, each component can have one or more inputs which are outputs from previous tasks. A task will not start unless its prerequisite task has completed successfully. Each task runs in a container that has been automatically created on a worker node in the cluster determined by Kubernetes according to the resource requirements for that task.

The advantage of using Kubernetes, apart from minimising the management of hardware, is that it is readily available in all major cloud services, such as Azure, AWS and GCP, as well as being installable on premises. What this means is that our machine learning pipeline solution is deployable with minimal changes on the previously mentioned cloud services as well as the existing hardware of our clients, provided Kubernetes can be installed on it. Running all the business logic in docker containers eliminates the need to install dependencies with every deployment. In fact, with a CI/CD pipeline we automatically build the docker images, run tests, and push the image to a container registry – internally and on the cloud. We build the images containing our code once and it can then be pulled many times from different locations.

Conclusion

In this article, we have described how machine learning is used in our pipeline to classify documents of various file types, such as PDF, Excel and CSV, and different image files. We have discussed how computer vision is used to identify the layout of a document page, and having done that, each constituent component is then further processed: figures and tables classified, and key terms identified from the extracted text. Furthermore, with the help of Kubernetes, documents can be processed at scale. By using this technology, we can achieve a more granular classification of the document contents and reduce project times significantly.

Acknowledgements

The authors would like to thank CGG for permission to publish this work. We would also like to thank our colleagues in the CGG Data Hub and AI Lab for the useful discussions.

References

Brown et al. [2020]. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, **33**, 1877-1901.

Devlin, J., Chang, M., Lee, K. and Toutanova, K. [2019]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT,* 4171-4186.

Koci E., Thiele M., Romero O. and Lehner W. [2019]. Cell Classification for Layout Recognition in Spreadsheets. Knowledge Discovery, Knowledge Engineering and Knowledge Management. IC3K 2016. *Communications in Computer and Information Science*, **914**, 78-100.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. [2012]. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, **25**.

Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z. and Zhou, M. [2020]. DocBank: A benchmark dataset for document layout analysis. *Proceedings of the 28th International Conference on Computational Linguistics,* 949-960.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. [2013]. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems,* **26**.

Pennington, J., Socher, R. and Manning, C. [2014]. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532-1543.

Ren, S., He, K., Girshick, R. and Sun, J. [2015]. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems,* **28**.

Tan, M. and Le, Q. [2019]. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proceedings of the 36th International Conference on Machine Learning*, 6105-6114.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. [2017]. Attention is All you Need. *Advances in Neural Information Processing Systems,* **30**.

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*, 1192-1200.

Zhong, X., Tang, J. and Yepes, A.J. [2019]. PubLayNet: largest dataset ever for document layout analysis. *2019 International Conference on Document Analysis and Recognition,* 1015-1022.