Intelligent Systems with Applications 20 (2023) 200285

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Automated invoice processing: Machine learning-based information extraction for long tail suppliers

Felix Krieger [*], Paul Drews, Burkhardt Funk

Institute of Information Systems, Leuphana University of Lüneburg, Germany

ARTICLE INFO

Keywords:
Layout-rich documents
Document analysis
Natural language processing

ABSTRACT

Automation of incoming invoices processing promises to yield vast efficiency improvements in accounting. Until a universal adoption of fully electronic invoice exchange formats has been achieved, machine learning can help bridge the adoption gaps in electronic invoicing by extracting structured information from unstructured invoice formats. Machine learning especially helps the processing of invoices of suppliers who only send invoices infrequently, as the models are able to capture the semantic and visual cues of invoices and generalize them to previously unknown invoice layouts. Since the population of invoices in many companies is skewed toward a few frequent suppliers and their layouts, this research examines the effects of training data taken from such populations on the predictive quality of different machine-learning approaches for the extraction of information from invoices. Comparing the different approaches, we find that they are affected to varying degrees by skewed layout populations: The accuracy gap between in-sample and out-of-sample layouts is much higher in the Chargrid and random forest models than in the LayoutLM transformer model, which also exhibits the best overall predictive quality. To arrive at this finding, we designed and implemented a research pipeline that pays special attention to the distribution of layouts in the splitting of data and the evaluation of the models.

1. Introduction

Invoices are essential documents for different business processes such as procurement and accounts payable. They hold the details of transactions between clients and suppliers, due to which they also bear legal value (Cristani et al., 2018) and are frequently used by external auditors as evidence for the existence of transactions and their correct recording (ISA 330, 2009, Krieger et al., 2021).

Invoicing is becoming progressively more digitalized, which increases the productivity and efficiency of the associated processes (Baviskar, Ahirrao, Potdar, et al., 2021, Cristani et al., 2018, Narayanam et al., 2020, Tanner & Richter, 2018). Electronic invoicing encompasses different degrees of digitization, from printed invoices, which are scanned over natively electronically readable invoices in PDF format that are exchanged by email, to invoicing as an integrated business-to-business process via electronic data interchange (EDI) (Tanner & Richter, 2018). In integrated business processes, the invoices may also be created automatically (Narayanam et al., 2020, 2021). Fig. 1 provides an overview of digital invoice processing, delineating the difference between the processing of unstructured invoice formats and EDI-based invoices.

PDFs contain information in an unstructured format and require an information extraction (IE) (Sarawagi, 2008) step to gain a semantic representation, as shown in Fig. 1. Invoices are, however, a particularly interesting type of business document, which makes the IE step simultaneously challenging and significant. While there are several pieces of information that can be expected to appear on an invoice, their positioning and schematical organization, the layout, is usually unregulated and to be freely decided on by the issuing company (Cristani et al., 2018).

In EDI-based invoicing, on the other hand, the information is directly delivered in a structured format, usually XML based, which facilitates any downstream processing (Narayanam et al., 2021, Tanner & Richter, 2018). Despite this advantage, the adoption of EDI invoicing is still progressing slowly, which can be attributed in part to the fact that companies benefit to varying degrees from its adoption (Tanner & Richter, 2018): Whereas companies receiving large quantities of invoices benefit heavily from EDI invoicing, the adoption barriers might be too great for companies for which this is not the case. Thus, companies looking to adopt EDI invoicing need to actively onboard their suppliers; they can achieve this onboarding most easily with suppliers from which they receive a large quantity of invoices. Infrequent suppliers, which make

* Corresponding author.
E-mail addresses: felix.krieger@leuphana.de (F. Krieger), paul.drews@leuphana.de (P. Drews), burkhardt.funk@leuphana.de (B. Funk).

https://doi.org/10.1016/j.iswa.2023.200285
Received 14 December 2022; Received in revised form 21 August 2023; Accepted 26 September 2023
Available online 2 October 2023

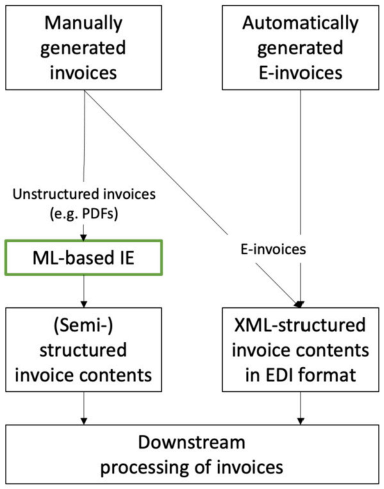
Manually generated invoices
Automatically generated E-invoices
Unstructured invoices (e.g. PDFs)
E-invoices
ML-based IE
(Semi-) structured invoice contents
XML-structured invoice contents in EDI format
Downstream processing of invoices

Fig. 1. Schematic depiction of digitalized invoice processing. The green box highlights the scope of our work.

up the biggest share of suppliers of most larger companies (Klein et al., 2004), are less inclined to adopt EDI invoicing, as they do not benefit to the same extent from its adoption (Tanner & Richter, 2018). IE from unstructured invoices therefore represents an attractive intermediate step toward automated invoice processing until EDI-based invoicing has universally superseded PDFs and paper-based invoices.

As mentioned before, the challenge for IE lies in the diversity of these long tail suppliers and the resulting variety of layouts (Cristani et al., 2018). Template-based IE solutions require a substantial amount of human involvement to create and maintain the different supplier templates, which can number hundreds (Cristani et al., 2018) or even thousands (Klein et al., 2004) and therefore offset the efficiency gains from automation. This shortcoming of template-based IE can be addressed through machine learning (ML).

A growing body of research proposes different ML approaches for IE from invoices, which would allow the training of models that are able to generalize to previously unseen invoice layouts. These approaches employ precomputed or learned structural representations that embed the visual and semantic properties of the invoices while preserving the layout. They can be broadly classified into graph-based (Krieger et al., 2021, Liu, Gao, et al., 2019, Lohani et al., 2019, Yu et al., 2020) and grid-based (Denk & Reisswig, 2019, Katti et al., 2018, Zhang et al., 2020, Zhao et al., 2019) approaches, as well as approaches relying on positional embeddings (Garncarek et al., 2021, Majumder et al., 2020, Xu et al., 2020, 2022). However, ML models are sensitive to the biases in the data being used for training (Shah et al., 2020). One crucial type thereof, selection bias, can originate from samples being used for training that are not representative of the population to which the model is then applied (Shah et al., 2020). Given the structure of suppliers present in many larger companies—few high-frequency suppliers, many low-frequency long tail suppliers— we are interested in understanding how the layout-aware ML approaches respond to training data taken from such a population. Therefore, the question that has motivated us to conduct this research is the following:

How do ML-based approaches to IE from invoices respond to skewed vendor distributions?

The contribution of this paper is twofold. First, we show that ML models can suffer from layout bias and that different models are affected to varying degrees by this bias. We train and evaluate different ML-based approaches to IE from a set of invoices that are skewed toward a few suppliers. The evaluation results are then disaggregated into in-sample and out-of-sample layouts, showing that all models are more accurate[1] on in-sample layouts. We also contribute to the literature by conducting a benchmark of different layout-aware ML models over a common set of invoices using a common evaluation metric. The results show that the pretrained transformer model LayoutLM (Xu et al., 2020) outperforms all other models in our benchmark by a comfortable margin, especially for out-of-sample layouts. LayoutLM is also the most robust against layout bias. The results further indicate that the other models in our study have different strengths with respect to detecting certain information entities.

The remainder of this paper is structured as follows: In section 2, we review relevant previous research and delineate the research gap. In section 3, we describe the employed methods and materials, after which the details of the model implementation are provided (section 4). Thereafter, we present (section 5) and discuss (section 6) the results, and we conclude our research in section 7.

2. Related literature

Due to the large possible varieties of layouts in invoices, they are considered a particularly challenging and interesting case for automated business document processing (Cristani et al., 2018). In essence, any form of automation requires the information contained on invoices to be made available for downstream applications in a structured fashion.

2.1. Challenges associated with the automated processing of invoices

The information on invoices can be broadly classified into two types of entities: header fields and line items. Header fields encompass, inter alia, the invoice number, the issue date, the total amount due, and the value-added-tax identification number (VAT ID) of the supplier (Cristani et al., 2018). Line items provide the details of the exchanged goods or services, such as a description of the provided good or service, the unit price, the number of units provided, and the total amount due per line item (Katti et al., 2018). EDI-based electronic invoicing facilitates the capture of invoice data by directly providing the information in a structured, usually XML-based, format (Tanner & Richter, 2018). However, the adoption of EDI invoicing remains limited (Koch, 2017, Tanner & Richter, 2018). Its main beneficiaries are large companies that receive large quantities of invoices. For many of their smaller suppliers, the incentive is weaker in the face of the organizational (Tanner & Richter, 2018) and technical (Cristani et al., 2018, Tanner & Richter, 2018) obstacles associated with the adoption of EDI-based invoicing. According to Koch (2017), even several years after adopting an EDI invoicing system, only 25%–30% of the invoices received by most large companies are fully electronic, and those they do receive are usually from business partners with whom they share a large number of transactions. This, however, represents only a small proportion of an organization's suppliers, as typically most suppliers only send invoices infrequently (Koch, 2019). The distribution of invoices received per supplier is therefore skewed toward a few high-frequency suppliers; for this reason, Tanner and Richter (2018) refer to lower-frequency suppliers, usually small and medium-sized enterprises, as the long tail. Fig. 2 provides a visual example of this distribution. This adoption gap can be bridged by technologies that allow the extraction of information from unstructured formats such as document images or PDFs.

2.2. Machine learning for IE from invoices

Early systems proposed for IE from business documents relied on manually preconfigured layout templates and rules (Baviskar, Ahirrao, Potdar, et al., 2021, Cristani et al., 2018, Klein et al., 2004). Other systems leverage fuzzy string matching algorithms to extract information

[1] In this paper, we refer to the "accuracy" of an ML model as its predictive quality, not as the evaluation metric.

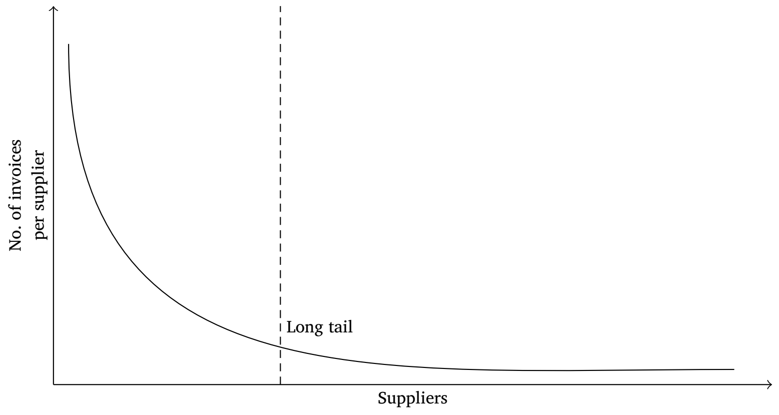

Fig. 2. Distribution of invoices per supplier as described by Koch (2019). Figure based on Koch (2019) and Tanner and Richter (2018).

through the triangulation between different documents (Datasnipper, 2023, Tater et al., 2022). A pioneering study applying ML for IE was presented by Palm et al. (2017). The authors introduced "CloudScan," a system that serializes the text of invoices into sequences of tokens and performs sequence tagging using a recurrent neural network to extract key-value pairs. CloudScan, however, dismisses the layout of the invoice documents, forcing the authors to decide on the ordering of the words (Palm et al., 2017). Subsequent approaches have addressed this shortcoming by developing document representations that retain the layout information. The approaches can broadly be classified into graph-based approaches (Liu, Gao, et al., 2019, Lohani et al., 2019, Yu et al., 2020), grid-based approaches (Denk & Reisswig, 2019, Katti et al., 2018, Zhao et al., 2019), and positional embeddings (Garncarek et al., 2021, Majumder et al., 2020, Xu et al., 2020, 2022).

Graph-based approaches model the document as a graph, in which text elements (words or paragraphs) are represented as nodes and the edges represent the spatial relationship between the text elements. The task of extracting key-value pairs is then modeled as node classification (Lohani et al., 2019, Yu et al., 2020, Zhang et al., 2020) or sequence tagging (Liu, Gao, et al., 2019), and graph convolution is employed to capture the contextual information between the text elements.

Grid-based approaches map the document text to a grid such that the relative spatial positions of the text elements are preserved. The grids can be of varying granularity: Chargrid (Katti et al., 2018) and BERTgrid (Denk & Reisswig, 2019) employ pixel-level grids, whereas CUTIE (Zhao et al., 2019) uses a coarser grid, in which each word is mapped to a single cell. The extraction of key-value pairs is then modeled as a semantic segmentation or object detection task (Denk & Reisswig, 2019, Katti et al., 2018, Zhao et al., 2019). Chargrid and BERTgrid also combine semantic segmentation with object detection to detect line item bounding boxes (Denk & Reisswig, 2019, Katti et al., 2018). The respective models capture contextual patterns through (grid) convolution mechanisms common in computer vision.

Another stream of study proposes models based on transformer architecture (Garncarek et al., 2021, Xu et al., 2020, 2022) and the self-attention-based mechanism (Majumder et al., 2020). The layout of the documents is accounted for by using learned (Majumder et al., 2020) or precomputed positional embeddings (Garncarek et al., 2021, Xu et al., 2022, 2020), which use the two-dimensional coordinates of the respective text's bounding box, analogical to the positional encodings proposed in Vaswani et al. (2017). The transformer-based models further employ pretraining strategies like BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) over a multitude of different layout-rich document types, aiming for a general understanding of documents. The pretrained models can then be fine-tuned for different downstream tasks over different types of layout-rich documents such as token classification to extract information from invoices.

2.3. Limits to comparability across different approaches to IE from invoices

All of the above mentioned studies address the problem of extracting information from invoices by using a different approach, in terms of either document representation, model architecture, or input features. Although they all present promising results, a direct comparison between approaches from the literature is hardly possible. Table 1 summarizes the used data sets, evaluation metrics, and evaluation baselines of the studies. It shows the basis on which the limited comparability between the proposed approaches is grounded.

First, most of the studies use some sort of proprietary set of invoices; this is because invoices often contain highly sensitive data and are therefore not made available to the general public (Baviskar, Ahirrao, & Kotecha, 2021, Baviskar, Ahirrao, Potdar, et al., 2021). The next reason is the granularity of the reported results: If results are reported for individual entity types, there is no established common set of entity types to be extracted from the invoices. This primarily affects line items: Only a few studies (Denk & Reisswig, 2019, Katti et al., 2018, Lohani et al., 2019) report the model's performance on line item entities.

A further limitation is given through the evaluation methodology. As can be seen in Table 1, the performance of the models is evaluated on different levels of granularity. The evaluations range from character-level (Denk & Reisswig, 2019, Katti et al., 2018) through word-level (Lohani et al., 2019) to entity-level (Garncarek et al., 2021, Xu et al., 2020, 2022, Yu et al., 2020). Most studies use established measures in information retrieval such as precision, recall, F_1 score, and average precision to evaluate the performance of their approach. With respect to the different levels of evaluation, however, it is not always clearly stated how the results are aggregated from lower-level predictions to higher-level measures. For example, the transformer-based models LAMBERT, LayoutLM and LayoutLMv2 employ byte-pair-encoding (BPE) (Sennrich et al., 2016) and WordPiece (Wu et al., 2016) subword tokenization, which subsequently yield subword-level predictions. Here, only Garncarek et al. (2021) state the use of the geometric mean as an aggregation method to arrive at entity-level measures from the predictions. In the case of Chargrid and BERTgrid, the evaluation is only conducted at the character level and not further aggregated at all; the authors use an accuracy measure similar to the word error rate (Denk & Reisswig, 2019, Katti et al., 2018). Due to the use of this metric, Chargrid and BERTgrid may not be compared to the other models that have not been used as evaluation baselines in the respective studies.

The use of baselines is another factor affecting the comparability between approaches. Especially in experiments where proprietary data sets have been used (Denk & Reisswig, 2019, Katti et al., 2018, Liu, Gao, et al., 2019, Lohani et al., 2019, Majumder et al., 2020, Yu et al., 2020, Zhao et al., 2019, Zhang et al., 2020), the authors resort to using a sequential model after the example of CloudScan and variants of their own models. Only a few studies use non-sequential approaches as baselines; in addition, Denk and Reisswig (2019) benchmark their

Table 1
Achieved results reported in the related literature along with the used data sets, evaluation metrics and baselines. Proprietary data sets are indicated with the abbreviation (pr.)

Model (Reference)	Data set(s)	Evaluation metrics	Reported aggregated results	Baselines
BERTgrid (Denk & Reisswig, 2019)	Invoices (pr.)	Character-level accuracy (Word-error rate) for three header entities and one line item entity, as well as aggregated over all entities and line item entities	65.48% ±0.58	Chargrid, Wordgrid (similar to CUTIE), variants of the proposed model
LAMBERT (Garncarek et al., 2021)	Mix of open and proprietary data sets for training, open data sets for evaluation (SROIE, CORD)	Entity-level F_1 scores averaged across entities for SROIE and CORD	94.41 F_1 on CORD, 98.17 F_1 on SROIE	RoBERTa (Liu, Ott, et al., 2019), LayoutLM, LayoutLMv2
Chargrid (Katti et al., 2018)	Invoices (pr.)	Character-level accuracy (Word-error rate) for five header entities and three line item entities	61,99% Macro average computed from the results reported for the individual entity types	Sequential (Bidirectional GRUs), variants of the proposed model
GAT+BiLSTM-CRF (Liu, Gao, et al., 2019)	Invoices (pr.), receipts (pr.)	F_1 score for six header entities for the invoice data set, averaged F_1 scores across all entities for invoices and receipts	0.873 F_1 on invoices, 0.836 F_1 on receipts	Sequential (BiLSTM+CRF)
GCN (Lohani et al., 2019)	Invoices (pr.)	Word-level F_1 score, precision, recall for 36 entities, including line item entities	0.93 F_1 (micro) on invoices	None
Self-attention (Majumder et al., 2020)	Invoices (pr.), receipts (SROIE)	F_1 score, ROC-AUC for seven resp. two header entities for invoices and receipts	0.878 F_1 (macro) on invoices	Variants of the proposed model
LayoutLM (Xu et al., 2020)	Open data sets for training and evaluation (incl. SROIE)	Entity-level precision, recall, F_1 score for SROIE	0.9524 F_1 on SROIE	Previous best on the SROIE leaderboard
LayoutLMv2 (Xu et al., 2022)	Open data sets for training and evaluation (incl. SROIE)	Entity-level precision, recall, F_1 score for SROIE	0.9601 F_1 on CORD, 0.9781 F_1 on SROIE	BERT (Devlin et al., 2019), UniLMv2 (Bao et al., 2020), LayoutLM (Xu et al., 2020); SROIE leaderboard, including PICK (Yu et al., 2020) and TRIE (Zhang et al., 2020)
PICK (Yu et al., 2020)	Medical invoices (pr.), train tickets (pr.), receipts (SROIE)	Mean entity precision (mEP), recall (mER), and F score (mEF) for six header entities on the medical invoice data set, as well as aggregated mEF on the train ticket and SROIE data sets	98.6 mEF on train tickets, 96.1 mEF on SROIE, 87.0 mEF on medical invoices	Sequential (BiLSTM+CRF), LayoutLM (Xu et al., 2020) (only on SROIE)
CUTIE (Zhao et al., 2019)	Receipts (pr.; SROIE)	Average precision (AP) and soft AP (tolerance for false positives) across keys for different receipt types, including SROIE	94.0, 81.5, 74.6 AP on different receipt type subsets	Sequential (Palm et al., 2017), BERT (Devlin et al., 2019), variants of the proposed model
TRIE (Zhang et al., 2020)	Taxi invoices (pr.), receipts (SROIE), resumes (pr.)	The quantity of goods or services provided. In case of services, this might relate to temporal units.	93.26 F_1 on taxi invoices, 96.18 F_1 on SROIE, 76.3 F_1 on resumes	Chargrid, Sequential (Ma & Hovy, 2016), GAT+BiLSTM-CRF (Liu, Gao, et al., 2019)

proposed model against a variant of CUTIE, and Zhang et al. (2020) use Chargrid and the graph attention-based model proposed in Liu, Gao, et al. (2019) as baselines.

In conclusion, the utilization of proprietary data sets, different evaluation methodologies and metrics, and the inconsistent use of baselines lead to poor comparability of different approaches in this field of research.

2.4. *Open data sets for IE from business documents and previous benchmarking studies on IE from invoices*

A partial remedy to the points outlined above is offered by the scanned receipts OCR and information extraction (SROIE) challenge (Huang et al., 2019) and the consolidated receipt data set for post-OCR parsing (CORD) (Park et al., 2019), as well as the Kleister data sets (Stanisławek et al., 2021), and the multi-layout invoice document data set (MIDD) (Baviskar, Ahirrao, & Kotecha, 2021).

SROIE and CORD are composed of receipts. Both data sets are similar in size; SROIE contains 973 receipts, of which 347 are dedicated to testing. CORD contains 1000 receipts, with 800 examples dedicated to training and 100 each for validation and testing. The SROIE data set is annotated for four header entity types: company, address, receipt date, and total amount. The CORD data set, on the other hand, encompasses only line item–like entities; most header fields have been left out due to privacy concerns (Park, 2021).

Stanisławek et al. (2021) introduce two data sets, Kleister-NDA and Kleister-Charity, which exhibit a greater complexity stemming from the multipage nature of the underlying document types. The data sets feature two types of layout-rich documents, nondisclosure agreements (Kleister-NDA) and financial statements (Kleister-Charity), which are annotated for several information entities. The MIDD data set is the only open data set containing invoices. It encompasses the OCR outputs for 630 invoices from four different layouts, annotated for seven invoice header fields. However, MIDD provides only the text and the corresponding annotations without the associated bounding box coordinates, due to which the layout information is lost.

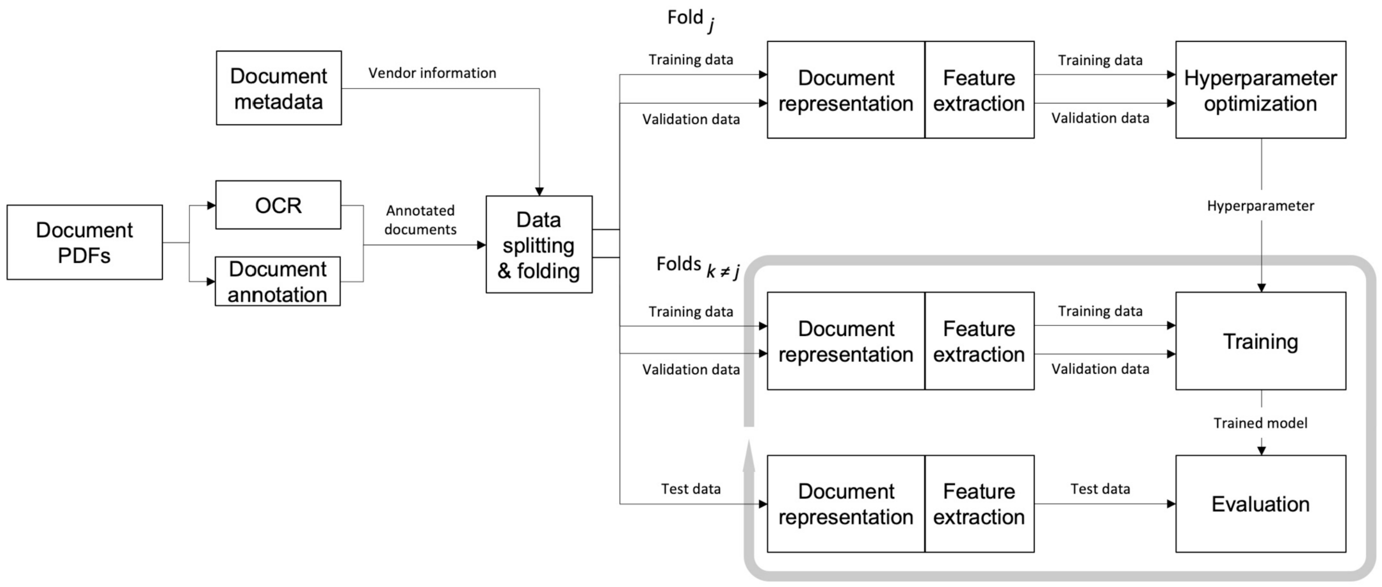

Fig. 3. Study pipeline. A set of invoices is passed through an optical character recognition (OCR) engine and annotated for several information entities. The annotated OCR outputs are used to train and evaluate different ML models over several iterations (folds). One fold is set aside for hyperparameter optimization.

SROIE, CORD, and the Kleister data sets are split into predefined train, validation, and test sets. For the SROIE, CORD, and Kleister-NDA data sets, there is no information provided on the quantity of the different layouts and their distribution in the respective splits, and for Kleister-Charity, the information is only rudimentary (Huang et al., 2019, Park et al., 2019, Stanisławek et al., 2021). Majumder et al. (2020) found that the receipts in the SROIE training and validation sets stem from 234 layouts, with 46 out of 626 receipts pertaining to one layout. As SROIE offers a public ladder in which researchers and practitioners may participate, the test set is kept private.

Introducing the Kleister data sets and Stanisławek et al. (2021) the benchmark sequential (Flair, (Akbik et al., 2019); BERT, (Devlin et al., 2019); RoBERTa, (Liu, Ott, et al., 2019)) and layout-aware models (LAMBERT, (Garncarek et al., 2021); LayoutLM, (Xu et al., 2020)) on the Kleister data sets shows that the layout-aware LAMBERT model outperforms all other models. Similar benchmarks have also been conducted on invoice data sets: Liu et al. (2016) constructed a comprehensive set of features and compare naive Bayes, logistic regression and support vector machine classifiers on a rather small data set composed of 97 invoices. In their study, the logistic regression and support vector machine classifiers outperformed the naive Bayes by a significant margin. Baviskar, Ahirrao, & Kotecha (2021) compared different word embedding techniques (Word2Vec, GloVe, FastText, embedding layer) for a sequential BiLSTM model on a data set of 1646 invoices from eight suppliers. They found that embedding words through a dedicated embedding layer achieved the best results.

To conclude, the studies proposing approaches to IE use mostly proprietary sets of invoices. The distribution of different layouts in the respective data sets is widely left unaddressed. The same applies to the open data sets that are being used to benchmark IE on visually rich documents. This leaves doubt as to whether the distribution of invoice layouts has an effect on the models. In addition, obtaining a comparison of the predictive quality of different layout-aware models is difficult due to the utilization of different evaluation metrics and the reliance on sequential models as baselines. Previous benchmarking studies on IE from invoices employ only either very small data sets (Liu et al., 2016) or data sets with small numbers of invoice layouts Baviskar, Ahirrao, & Kotecha (2021). Furthermore, neither study employs any of the layout-aware approaches introduced in section 2.2. We therefore see the need for a study that benchmarks different approaches to IE from invoices and explicitly addresses the distribution of layouts.

3. Study design

The goal of this study is to close the gaps mentioned in the previous section by conducting an independent benchmark study of different ML-based approaches to information extraction from invoices. We devised a pipeline similar to the pipelines employed by Baviskar, Ahirrao, & Kotecha (2021), Liu et al. (2016), and Stanisławek et al. (2021), which is pictured in Fig. 3. The main differences between our pipeline and the one presented in previous research (Baviskar, Ahirrao, & Kotecha, 2021, Liu et al., 2016, Stanisławek et al., 2021) are that we specifically accounted for suppliers—and therefore layouts—in the data splitting and model evaluation. Furthermore, we tuned the neural networks for a determined set of hyperparameters to achieve the best possible accuracy for each model. This section provides details on the data set and the methodology utilized for annotating the data set, tuning and training the models, and evaluating their accuracy.

As there are only few implementations available of the abovementioned models, we had to limit our study to a few selected examples. Our goal is to represent a broad range of methodologies; hence, we have chosen examples for each of the three model types described in section 2.2: graph based, grid based and transformer based. We chose Chargrid and BERTgrid as grid-based approaches, and the GCN- and GAT-based models introduced in Lohani et al. (2019) and Liu, Gao, et al. (2019). All of them have been specifically designed for invoice documents. We also included the LayoutLM transformer model, whose authors aimed for a more general understanding of business documents to see how it compares to invoice-specific approaches. In addition to the rather complex neural network models, we included a random forest model in our evaluation as a baseline.

3.1. Data set

Our data set was composed of 1059 invoices provided by a large German firm that sources products and services from a multitude of countries across the world. The predominant language of the documents was English (955 invoices), followed by German (76 inv.). The less frequent languages were French (8 inv.), Dutch (8 inv.), Spanish (7 inv.) and Italian (5 inv.). The invoices in our data set documented the procurement of physical goods as well as services. Out of the 1059 invoices, 63 were multipage documents, resulting in a total of 1126

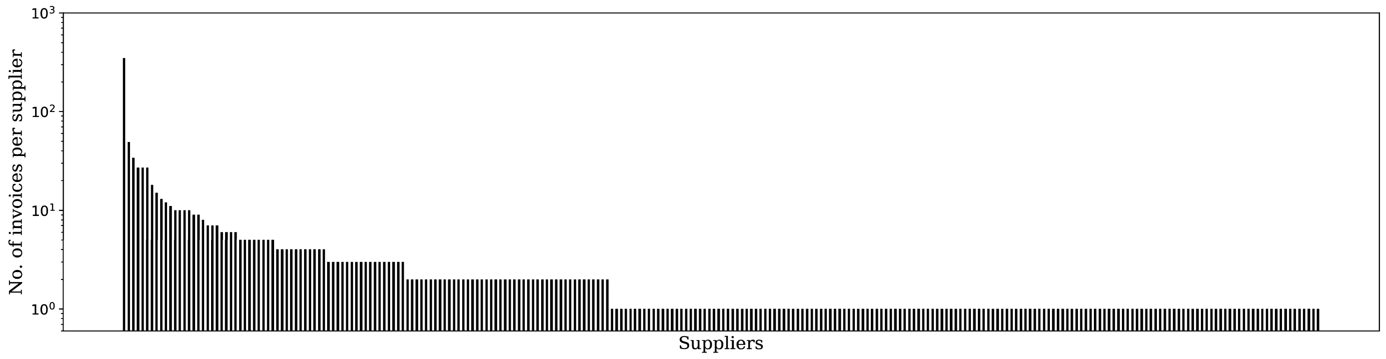

Fig. 4. Barplot of the supplier distribution in the data set; each bar represents one supplier. The distribution follows the pattern described by Koch (2019).

invoice pages. The invoices stemmed from 259 different suppliers, reflecting a wide variety of invoice layouts. The distribution of suppliers was, however, skewed. Fig. 4 plots the distribution of suppliers in the data set. The figure shows that the distribution of suppliers follows a long-tailed distribution; a large subset of invoices stemmed from a handful of suppliers, including one supplier who was disproportionately represented with 348 invoices. While the associated invoices followed the same layout structure, they still exhibited variance between them; the documents could contain different counts of line items, which in turn may be of varying length.

3.2. Data set annotation

We extracted the text from the abovementioned invoices using an optical character recognition (OCR) engine. The OCR step yielded 249,032 word-level text boxes from the 1126 invoice pages. The documents were annotated by two domain experts for 14 different information entities. The annotation was done by drawing bounding boxes over the document images using Microsoft Azure Machine Learning's labeling tool for object detection. A corresponding class label was assigned to a text box if its area was at least 80% covered by an annotation bounding box. We further assigned a background class to all text boxes that were not sufficiently covered by an annotation box. In total, 54,023 text boxes were assigned an entity class, with recipient address being the most frequently assigned class (13,683 annotated text boxes) and total tax amount the least frequently assigned class (383 annotated text boxes). The data set therefore exhibits a sharp class imbalance, both between the background class and the information entity classes and between the entity classes themselves. Note that some entity types did not appear on every invoice, whereas others could appear multiple times on the same invoice, as was the case with invoice numbers. Table 7 in the appendix provides the details on the assigned entity classes, along with the data types for each entity class. The data types were also used later as features for the random forest and GCN models. This section and the previous section outline two biases in the data set: a skewed distribution of suppliers and a steep class imbalance. To gain a valuable assessment of the models' accuracy, they must be accounted for during training and evaluation.

3.3. Evaluation

Given the characteristics of the data set, an appropriate evaluation methodology was required to accurately address the research question. The skewed supplier distribution called for an according stratification when the data were split into the respective training, validation, and test sets. On the other hand, the evaluation metric should have been unaffected by the class imbalance.

3.3.1. Data splitting and folding strategy

As we wanted to assess the accuracy of the models on both invoices whose layouts have been part of the training set and invoices whose

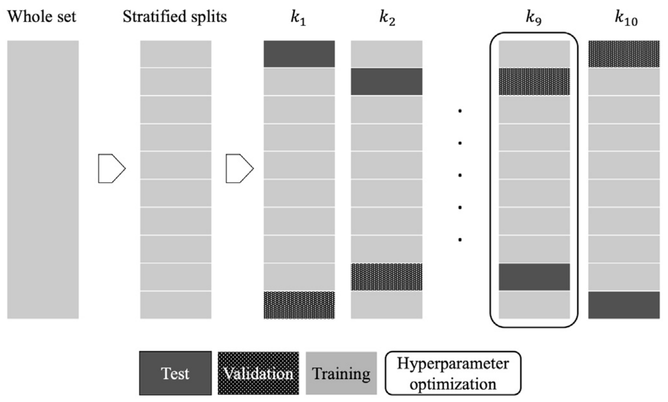

Fig. 5. The invoices are split into 10 subsets. Each box represents one split; the colors indicate its utilization as training, validation, or testing data in the different folds. The tuning fold is set aside for hyperparameter optimization.

layouts the models have not been exposed to, we ensured that each split contained a proportion of invoices from suppliers that were exclusive to it. In this way, we maximized the variance of invoice layouts in the training set while simultaneously enabling a decomposition of the evaluation results on the respective invoices with in-sample and out-of-sample layouts. To address the stochastic influences governing the data splitting, a 10-fold cross-validation (Han et al., 2011) was employed. This also allowed us to judge the robustness of the models toward varying training and testing data. Fig. 5 provides a graphical representation of the folding strategy. The documents in the data set were split into 10 (approximately) equally sized, mutually exclusive subsets of invoices. We then generated 10 different folds from the subsets so that each split was used as the test set once. Fig. 6 shows the distribution of suppliers per split and highlights the suppliers contributing the documents with out-of-sample layouts. To prevent the models from overfitting to the training data, one split in each fold was assigned as the validation set, on which a stopping criterion was applied during the training (see section 3.4). One split was set aside as the validation set for the hyperparameter optimization (see section 3.5) and subsequently discarded from the evaluation; the evaluation metrics were therefore computed over 9 folds.

3.3.2. Evaluation metric

To assess the model performance during training and for the final evaluation, we employed the F_1 score, similar to Stanisławek et al. (2021). The score is commonly used in information retrieval and is well suited to judging the predictive quality of a model in the presence of unbalanced class distributions (Manning et al., 2008). The models were evaluated at the text-box level against the ground truth described in section 3.2. For the models returning a probability distribution P over the entity types Y, we selected the entity type with the highest probability as the prediction. If the model yielded more granular predictions

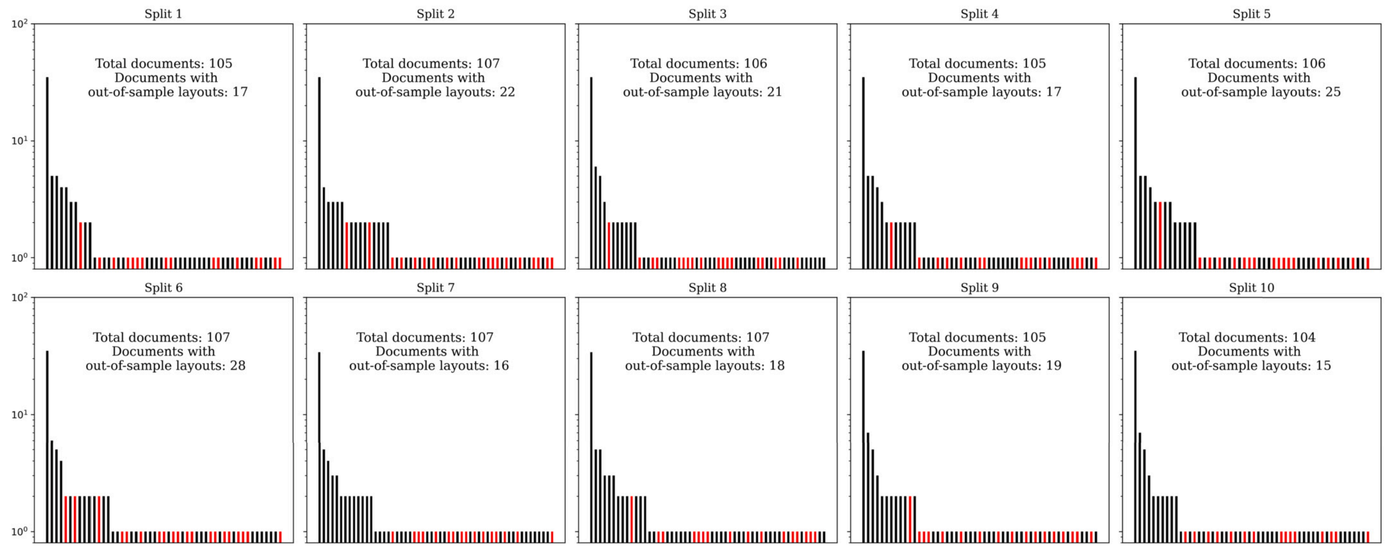

Fig. 6. Bar charts of the supplier distribution in the individual splits analogous to Fig. 4. The red bars indicate suppliers that do not appear in the training set (out-of-sample suppliers) if the respective split is used as the test set.

Table 2
Models selected for benchmarking.

Model	Layout representation	Input features	Information extraction	Prediction granularity
BERTgrid	Grid	BERT embeddings	Semantic segmentation, object detection	Pixel-level
Chargrid	Grid	One-hot-encoded characters	Semantic segmentation, object detection	Pixel-level
GCN	Word-level graph	BPE, manually constructed features	Node classification	Word-level
GAT+ BiLSTM-CRF	Paragraph-level graph	Word2Vec (Mikolov et al., 2013) embeddings, manually constructed distance-related edge features	Sequence tagging, node classification	Word-level
Lay-outLM	Positional embedding	Tokens	Word classification	Token-level
Random Forest	-	Manually constructed text box and string features	Word classification	Word-level

j associated with one text box i, such as n subword tokens or pixels, the predictions for each text box were aggregated using the geometric mean, as has been used by Garncarek et al. (2021):

$$\hat{y}_i = arg\,max_y \left(\prod_{j=1}^{n} P(Y=y)_{i,j} \right)^{\frac{1}{n}}$$

This way, we ensured the comparability of results across different models that yield predictions of differing granularity (see Table 2).

3.4. Training

During training, an early stopping criterion kept the models from overfitting to the training data (Bengio, 2012). We monitored the prediction-level macroaveraged F_1 score, the unweighted mean of the F_1 scores per entity, on the validation set with a patience of 10 epochs. The macroaveraged F_1 score included the background class, as the models needed to be able to distinguish between relevant entities and background text. Where applicable, we followed the class-weighting scheme proposed by Paszke et al. (2016) to address the class imbalance described above, similar to Katti et al. (2018):

$$w_{class} = \frac{1}{ln(c + p_{class})}$$

Using the class frequency p_{class}, the scheme yielded static class weights w_{class}, which affected how much the examples of each class affected the training loss; the greater the weight, the more a falsely predicted instance of the associated class contributed to the loss. The class weights were governed by a single hyperparameter c, which could be easily tuned for during hyperparameter optimization.

3.5. Hyperparameter optimization

The performance of neural networks is significantly tied to the chosen hyperparameters (Smith, 2018). We therefore optimized their hyperparameters using Bayesian optimization (Snoek et al., 2012), aiming to maximize the prediction-level F_1 score on the validation set. The optimization was performed with one dedicated split as the validation set (see section 3.3), to which the fold using the same split as the test set is excluded from the evaluation. Following the recommendations given by Bengio (2012), when working with limited resources, we focused on the learning rate α and the minibatch size B. We further tuned for a third hyperparameter: the class weighting factor c, which governs the distribution of class weights (see section 3.4). As the models and data representations consumed varying amounts of GPU memory, we determined the sample interval for B for each model by starting with $B_1 = 2$ and exponentially increasing B until the GPU memory limit was reached. The sample interval for α was determined using a α-range test (Smith, 2017), exponentially increasing α from 10^{-8} to 1.0 using the maximum previously determined B. For c, we chose a sample interval between 1.0 and 2.0. A factor of 1.0 completely leveled out any imbalances between the classes, whereas 2.0 assigned each class identical weights, leaving the class distribution untouched.

4. Implementation

We developed a research environment in Python using different ML packages and frameworks, mainly around the PyTorch ecosystem. We used PyTorch 1.7.0 with CUDA 10.1 to implement, train, and test all

Table 3
Tuned hyperparameters per model.

	α		B		c	
Model	Search interval	Selected	Search interval	Selected	Search interval	Selected
BERTgrid	$\{2.7542\times10^{-4}, 2.7542\times10^{-2}\}$	1.4550×10^{-2}	{2, 4}	2	{1.0, 2.0}	1.2292
Chargrid	$\{2.2909\times10^{-4}, 2.2909\times10^{-2}\}$	2.1209×10^{-2}	{2, 4}	4	{1.0, 2.0}	1.1319
GCN	$\{1.9055\times10^{-6}, 1.9055\times10^{-4}\}$	1.9055×10^{-4}	{2, 64}	16	{1.0, 2.0}	1.2149
GAT+BiLSTM-CRF	$\{3.0200\times10^{-5}, 3.0200\times10^{-3}\}$	9.8624×10^{-4}	{2, 16}	16	-	-
LayoutLM	$\{1.6596\times10^{-6}, 1.6596\times10^{-4}\}$	3.2771×10^{-4}	{2, 8}	4	{1.0, 2.0}	1.2112

models and referred to PyTorch-geometric (Fey & Lenssen, 2019) for any graph layers in the models. The BERTgrid, Chargrid, GAT+BiLSTM-CRF, and GCN models were implemented directly by us; the implementations of LayoutLM and random forest were obtained from the huggingfaces transformers library (Wolf et al., 2020) and scikit-learn (Pedregosa et al., 2011). Tesseract 4.0 was employed as the OCR engine to extract the text boxes from the documents. It should be noted that Tesseract is able to extract paragraph-level text boxes, which, however, can be of arbitrary accuracy on invoices. We utilized this functionality of Tesseract for the GAT+BiLSTM-CRF model (see section 4.3). To streamline and execute the experiments of the PyTorch-based models, we employed PyTorch Lightning. All experiments were run on a single computing instance with 6 processor cores, 118 GB RAM, and an Nvidia K80 GPU with 8 GB video RAM. In this section, we provide the implementation details for the individual models. Table 3 gives an overview of the chosen hyperparameters that resulted from the optimization step on the neural network models. The following subsections provide the implementation details for the individual models.

4.1. Chargrid

Our implementation of Chargrid deviated in minor aspects from the description given in Katti et al. (2018). We did not oversample invoices with multiple line items during training—the data set was already quite small, with repeating invoice layouts, and we wanted to limit further risk of overfitting. We maintained the target resolution for the Chargrid representations used in the original paper. Furthermore, Katti et al. (2018) mention employing anchor boxes (Ren et al., 2015) to support the object detection task; their parametrization is, however, not further specified. Following the recommendations given by Zhang et al. (2021), we therefore decided to generate a total of 13 anchor boxes per pixel, using ten linearly increasing aspect ratios $\mathbf{r}$ (relation of anchor box width to height) from 5 to 50, $\mathbf{r} = [5, \ldots, 50]$ and four linearly increasing scales $\mathbf{s}$ from 0.1 to 0.4, $\mathbf{s} = [0.1, \ldots, 0.4]$. The chosen $\mathbf{s}$ and $\mathbf{r}$ form rectangular anchor boxes of flat and wide shape and allowed us to accommodate line items of varying sizes. To evaluate the model, we upsampled the predicted segmentation mask back to the document's original resolution by inverting the interpolation from the Chargrid construction step (Katti et al., 2018). The upsampled segmentation mask was then compared against the ground-truth's labeled text boxes.

4.2. BERTgrid

BERTgrid (Denk & Reisswig, 2019) employs the same model architecture as Chargrid, the main difference between the two being the document representation fed into the model. Whereas Chargrid represents a document by one-hot-encoding characters on the pixel level, BERTgrid embeds tokens using a pretrained BERT model. Our implementation deviated from the approach described in Denk and Reisswig (2019) in the usage of said BERT model: As we did not have a large quantity of invoices to pretrain a custom BERT model, we used a standard multilingual pretrained BERT model obtained from the huggingface transformers library (Wolf et al., 2020), which was not further pretrained on invoices. As the BERTgrid document representation yields large tensors and WordPiece tokens are generally coarser than characters, we downsampled the document representation to half the target resolution of our Chargrid representation. In all other aspects, our implementation of BERTgrid equalled our implementation of Chargrid.

Table 4
Results of the parameter search for the GAT+BiLSTM-CRF model.

Layer	Search interval	Selected
BiLSTM_1	[64, 2048]	512
GAT_1	[64, 2048]	256
GAT_2	[64, 2048]	256
BiLSTM_2	[64, 2048]	512

4.3. GAT+BiLSTM-CRF

The model described in Liu, Gao, et al. (2019) combines different mechanisms to model contextual relationships; graph-attention layers are combined with BiLSTM layers and a CRF sequence tagging head. For our implementation of the model, we opted for two graph-attention layers in light of the results of the ablation study presented in Liu, Gao, et al. (2019). A two-layer multiperceptron (MLP) with ReLu nonlinearity embeds the node-edge-node features in each graph layer. As the authors did not specify an optimizer for training in the original paper, we employed ADAM (Kingma & Ba, 2017), similar to Lohani et al. (2019) and Xu et al. (2020). The sizes for the model layers—that is, the number of hidden nodes per layer—were not given in the paper. They govern the complexity of the model and therefore the ability of the model to generalize (Bengio, 2009). Hence, we performed a bandit-based random parameter search with successive halving (Karnin et al., 2013) aimed at maximizing the macroaveraged F_1 score on the validation set. The same cross-validation iteration as for the hyperparameter optimization described above was used. The search ran for 50 iterations, each time training the model for 30 epochs. Table 4 gives the results of the parameter search. Note that the MLPs that embed the edge-to-edge features have the same number of nodes as the respective GAT layers. To build the underlying document graph as described by Liu, Gao, et al. (2019), we extracted paragraph-level OCR outputs, which can span text passages ranging from single words to multiple lines. For each word in the paragraph, we extracted word embeddings using pretrained Word2Vec (Mikolov et al., 2013) vectors via gensim (Řehůřek & Sojka, 2010). To enable batch-wise computation, we padded the length of all paragraphs to the length of the longest paragraph in our data set, which was 78 words. We assigned a distinct vector to each padding element and constructed a padding mask for each document indicating non-padding elements for the CRF layer. Out-of-vocabulary words were similarly assigned a designated embedding vector. When evaluating the model, the most likely sequence of tags for each paragraph was obtained using the Viterbi algorithm (Forney, 1973). The sequence was then converted from the IOB tagging scheme back to the original set of labels and compared against the ground truth. Note that each word in a paragraph corresponded to one text box in the ground truth.

Table 5
Results of the parameter search for the GCN model.

Layer	Search space	Selected
GCN_1	[64, 2048]	512
GCN_2	[64, 2048]	256
GCN_3	[64, 2048]	256
GCN_4	[64, 2048]	128

4.4. GCN

The model presented by Lohani et al. (2019) employs graph convolution over a precomputed document graph similar to Liu, Gao, et al. (2019); however, it relies solely on node classification to extract any relevant information pieces. Our implementation only deviated in a minor detail from the description provided in Lohani et al. (2019): As we did not have the author's knowledge- or data bases available, we did not implement any related features. The BPE embeddings were obtained from BPEmb (Heinzerling & Strube, 2018). As the description of the model architecture given by the authors did not include layer size specifications, we ran a parameter search in the same fashion as for the GAT+BiLSTM-CRF model. Table 5 contains the results.

4.5. LayoutLM

We obtained an implementation of the LayoutLM model with pretrained weights via the huggingfaces transformers library (Wolf et al., 2020), which corresponds to the LayoutLM$_{Base}$ configuration with text and layout modality, complemented with a token classification head composed of a two-layer MLP with dropout. As proposed by Xu et al. (2020), we fine-tuned the parameters of the whole (pretrained) model on the token classification task using an ADAM optimizer (Kingma & Ba, 2017). The learning rate was warmed up linearly for one epoch. When extracting the tokens for each document, we scaled the x and y of the textboxes to the range {0, 1000}. The extracted token sequences were then padded to the maximum length possible (512). If their length already exceeded the maximum length, we applied a sliding window approach.

4.6. Random forest

As described above, we also trained and evaluated a random forest model (Breiman, 2001). Random forests are known to be robust, with only very limited performance gains to be achieved via hyperparameter tuning (Probst et al., 2019). We therefore abstained from tuning the random forest and used the standard parameters. The model was trained on the OCR outputs, which had been enriched with several manually constructed features. One row in the data set corresponded to one single text box. The manual features captured the properties of the text associated with each text box, such as its length and the number of (special) characters and digits, and data types (see appendix A). We furthermore calculated dictionary-based features that searched for the presence of discriminative terms such as "Invoice no." or "Total" in the vicinity of the text box. These features mimicked the context diffusion mechanisms in the neural networks. Handling text boxes independently from each other further allowed us to employ subsampling to address the class imbalance. We subsampled the background class so that the numbers of labeled and unlabeled instances in the training set were equal.

5. Results

Using the implementation described above, we tuned, trained, and evaluated the models as described in section 3.3. We report the F_1 scores with standard deviation per model and entity type averaged across all folds and the corresponding macro averages in Table 6. The results are further disaggregated down into results obtained from invoices with in-sample and out-of-sample layouts. In the tables, **bold** and underlined results indicate the best and second-best results per entity class. Fig. 7 summarizes the tables and provides box plots of the attained F_1 scores per model. The figure shows a significant gap in predictive quality between in-sample and out-of-sample layouts, both in terms of macro average (green dashed line) and scattering. This gap—which we will refer to from here on as the accuracy gap—varies between the different models. While the accuracy gap of the LayoutLM model—the overall best model in the study—is 0.1742 points, it is much larger for the other models. The most affected are the random forest and Chargrid models, with gaps of 0.4299 and 0.4206 F_1, respectively. The GCN and BERTgrid modes are less affected: Their accuracy drops by 0.3287 and 0.3187, respectively. LayoutLM achieves the best overall results, both for in- and out-of-sample layouts in terms of macroaveraged F_1 scores: 0.8761, and 0.7019, respectively. Looking at the overall results and the results over in-sample layouts, its closest competitor is the random forest model, with macro in-sample and out-of-sample F_1 scores of 0.7744 and 0.8288, respectively. It outperforms all neural networks other than LayoutLM in terms of macroaveraged F_1, and it even manages to outperform LayoutLM in the detection of line item quantities over in-sample layouts. This picture changes as we look at the out-of-sample results. Random forest's accuracy drops to 0.3989, being subsequently outperformed by both the BERTgrid and the GCN models. While BERTgrid is slightly better than the GCN model over out-of-sample layouts, GCN achieves a lower standard deviation, indicating more consistent results. This observation also holds true for in-sample layouts and overall results. The overall worst-performing model is the GAT+BiLSTM-CRF model in terms of both macro average F_1 and standard deviation. However, its accuracy gap between in- and out-of-sample layouts is smaller than that of random forest and Chargrid. Comparing the overall results for the individual entity classes, all models extract the recipient address with the highest accuracy and the total tax amount with the lowest. However, when reduced to the out-of-sample layouts, the models show worse performance on the line item tax amounts than on the total tax amounts; even LayoutLM achieves an F_1 of merely 0.2826. This is surprising, considering that LayoutLM achieves a very good F_1 score of 0.8811 on the class over in-sample layouts. All of the models fail in at least one fold to extract the line item tax amounts; the GAT+BiLSTM-CRF does not succeed at all. Apart from LayoutLM, which almost always outperforms all other models, the models exhibit individual strengths in extracting certain entities. For example, the GAT+BiLSTM-CRF achieves the second-highest F_1 score for the vendor VAT ID on out-of-sample layouts, whereas Chargrid almost matches the accuracy of LayoutLM for line item tax amounts.

6. Discussion

The goal of our research was to determine how ML-based approaches to IE from invoices respond to a skewed distribution of suppliers that is characterized by few highly frequent suppliers and a long tail of infrequent suppliers. Specifically, we were interested in the accuracy of the models on the invoices of long tail suppliers.

Our research approach pays special attention to the distribution of layouts in the training, validation, and test data sets, which has been largely neglected in previous research on IE from invoices. The data in our study are split into mutually exclusive subsets such that each subset contains a proportion of out-of-sample layouts issued by long tail suppliers. We found that all models in our study were significantly less accurate on out-of-sample layouts than on in-sample layouts. This is reflected in both lower macro F_1 scores and higher standard deviations between folds.

These findings suggest that there exists a layout bias of which designers of IE systems for invoices need to be aware. If we had not disaggregated the classification results into in-sample and out-of-sample layouts, the accuracy gap between them could have gone undetected.

Table 6
F_1 scores with standard deviation by model and entity class across **in-sample** and **out-of-sample** layouts in the test sets.

	All layouts						In-sample layouts						Out-of-sample layouts					
Entity type	BERTgrid	Chargrid	GAT+ BiLSTM-CRF	GCN	LayoutLM	Random Forest	BERTgrid	Chargrid	GAT+ BiLSTM-CRF	GCN	LayoutLM	Random Forest	BERTgrid	Chargrid	GAT+ BiLSTM-CRF	GCN	LayoutLM	Random Forest
Invoice Number	0.8122 (0.0292)	0.7624 (0.0380)	0.5797 (0.1620)	0.7761 (0.0336)	0.9115 (0.0252)	0.8505 (0.0284)	0.8989 (0.0364)	0.9103 (0.0520)	0.7600 (0.2542)	0.8578 (0.0647)	0.9324 (0.0315)	0.8938 (0.0474)	0.3830 (0.1005)	0.2104 (0.0754)	0.2278 (0.1073)	0.4203 (0.0914)	0.7476 (0.0985)	0.4762 (0.1127)
Issue Date	0.8702 (0.0414)	0.8157 (0.0708)	0.6953 (0.2140)	0.8291 (0.0463)	0.9241 (0.0244)	0.8177 (0.0447)	0.8827 (0.0327)	0.8826 (0.0344)	0.6465 (0.2057)	0.8469 (0.0371)	0.9465 (0.0258)	0.9150 (0.0266)	0.6028 (0.1257)	0.3237 (0.1067)	0.3972 (0.2354)	0.5914 (0.1212)	0.8376 (0.0673)	0.4884 (0.0933)
Total Amount	0.8130 (0.0477)	0.7857 (0.0398)	0.5539 (0.1755)	0.7309 (0.0575)	0.8288 (0.0555)	0.7394 (0.0582)	0.9137 (0.0435)	0.8820 (0.0693)	0.7304 (0.2611)	0.8757 (0.0479)	0.9407 (0.0257)	0.8843 (0.0460)	0.6262 (0.0988)	0.4966 (0.0566)	0.2410 (0.1428)	0.4861 (0.1255)	0.7302 (0.0886)	0.4693 (0.0958)
Vendor VAT ID	0.5933 (0.0535)	0.6155 (0.0591)	0.5611 (0.1819)	0.7317 (0.0479)	0.8563 (0.0359)	0.7363 (0.0505)	0.8154 (0.0391)	0.8137 (0.0364)	0.6943 (0.2090)	0.6823 (0.0644)	0.8609 (0.0313)	0.7343 (0.0394)	0.2850 (0.1448)	0.0797 (0.0690)	0.3639 (0.1819)	0.2641 (0.0833)	0.7068 (0.1082)	0.2059 (0.1247)
Due Date	0.8536 (0.0313)	0.8444 (0.0638)	0.7156 (0.2169)	0.7822 (0.0697)	0.9097 (0.0257)	0.8275 (0.0496)	0.8323 (0.0547)	0.8299 (0.0524)	0.6095 (0.1708)	0.8210 (0.0415)	0.8636 (0.0373)	0.8418 (0.0471)	0.4140 (0.2306)	0.1374 (0.1645)	0.1945 (0.2173)	0.2609 (0.1918)	0.7578 (0.1092)	0.1928 (0.1555)
Provision Date	0.7045 (0.0337)	0.5971 (0.0428)	0.4468 (0.1284)	0.6008 (0.0753)	0.7822 (0.0554)	0.6407 (0.0601)	0.7817 (0.0991)	0.8610 (0.0680)	0.6362 (0.1785)	0.8563 (0.0358)	0.9097 (0.0221)	0.8879 (0.0334)	0.6077 (0.1073)	0.1514 (0.1487)	0.2110 (0.1463)	0.4522 (0.1881)	0.6916 (0.1024)	0.3873 (0.2043)
Recipient Address	0.9361 (0.0135)	0.9344 (0.0199)	0.8396 (0.2321)	0.9136 (0.0126)	0.9602 (0.0094)	0.8630 (0.0174)	0.8718 (0.0306)	0.8778 (0.0343)	0.6784 (0.2055)	0.8277 (0.0327)	0.8956 (0.0420)	0.8822 (0.0297)	0.8004 (0.0662)	0.7850 (0.0825)	0.7761 (0.0861)	0.8048 (0.0457)	0.9066 (0.0282)	0.5795 (0.0936)
Vendor Address	0.5619 (0.0471)	0.5133 (0.0339)	0.7523 (0.2155)	0.7801 (0.0448)	0.8897 (0.0326)	0.7838 (0.0247)	0.7394 (0.0454)	0.7618 (0.0417)	0.5765 (0.2285)	0.7452 (0.0368)	0.7775 (0.0574)	0.7686 (0.0317)	0.6137 (0.0546)	0.3752 (0.1018)	0.5163 (0.1096)	0.5142 (0.0827)	0.7999 (0.0589)	0.4072 (0.0849)
Total Tax Amount	0.5464 (0.0997)	0.6206 (0.0673)	0.3626 (0.1741)	0.6036 (0.0837)	0.6917 (0.0544)	0.6912 (0.0681)	0.7279 (0.0411)	0.6804 (0.0542)	0.4966 (0.1523)	0.6338 (0.0772)	0.8005 (0.0675)	0.6904 (0.0497)	0.2931 (0.1994)	0.3846 (0.2143)	0.1019 (0.1398)	0.4133 (0.1196)	0.5442 (0.0441)	0.3106 (0.2080)
Line Item Description(s)	0.7797 (0.0321)	0.7708 (0.0322)	0.6400 (0.1764)	0.6413 (0.0543)	**0.8317** (0.0270)	0.6851 (0.0216)	0.9579 (0.0138)	0.9588 (0.0150)	0.8487 (0.2650)	0.9321 (0.0104)	**0.9692** (0.0089)	0.9155 (0.0194)	0.6232 (0.1022)	0.5784 (0.1000)	0.3967 (0.1120)	0.4704 (0.0677)	**0.6967** (0.0868)	0.5173 (0.0427)
Line Item Price(s)	0.7857 (0.0462)	0.7903 (0.0507)	0.5727 (0.1596)	0.7908 (0.0378)	**0.8441** (0.0318)	0.7925 (0.0509)	0.8460 (0.0560)	0.8427 (0.0511)	0.6064 (0.2198)	0.7754 (0.0633)	**0.8513** (0.0621)	0.7886 (0.0632)	0.3275 (0.1271)	0.4020 (0.1770)	0.0951 (0.1234)	0.5452 (0.1294)	**0.6840** (0.1156)	0.4276 (0.0880)
Line Item Quantity(s)	0.7491 (0.0943)	0.8381 (0.0692)	0.5999 (0.1651)	0.8228 (0.0436)	**0.8938** (0.0197)	0.8456 (0.0292)	0.5974 (0.1320)	0.6703 (0.0427)	0.4089 (0.1989)	0.6458 (0.0917)	0.7284 (0.0609)	**0.7594** (0.0683)	0.2738 (0.1993)	0.5154 (0.1948)	0.2182 (0.1299)	0.5410 (0.1715)	**0.7262** (0.1006)	0.4447 (0.1621)
Line Item Subtotal(s)	0.8360 (0.0324)	0.8352 (0.0295)	0.6184 (0.1884)	0.7881 (0.0278)	**0.8665** (0.0425)	0.8237 (0.0314)	0.5493 (0.0520)	0.5376 (0.0345)	0.7983 (0.2423)	0.8310 (0.0490)	**0.9080** (0.0347)	0.8553 (0.0246)	0.6237 (0.1235)	0.5899 (0.0722)	0.2149 (0.1383)	0.5942 (0.1112)	**0.7154** (0.1037)	0.5257 (0.0985)
Line Item Tax Amount(s)	0.7226 (0.0424)	0.7445 (0.0403)	0.5564 (0.2284)	0.7237 (0.0389)	**0.7540** (0.0584)	0.7449 (0.0362)	0.6324 (0.0593)	0.6785 (0.0644)	0.5946 (0.2442)	0.7962 (0.0507)	**0.8811** (0.0326)	0.7861 (0.0483)	0.1111 (0.2357)	0.2696 (0.2000)	0.0000 (0.0000)	0.1675 (0.1643)	**0.2826** (0.2224)	0.1515 (0.1743)
Macro Avg.	0.7546 (0.1241)	0.7477 (0.1231)	0.6067 (0.2141)	0.7511 (0.0982)	**0.8532** (0.0791)	0.7744 (0.0786)	0.7891 (0.1334)	0.7991 (0.1228)	0.6490 (0.2363)	0.7948 (0.1000)	**0.8761** (0.0776)	0.8288 (0.0816)	0.4704 (0.2360)	0.3785 (0.2336)	0.2825 (0.2346)	0.4661 (0.1979)	**0.7019** (0.1727)	0.3989 (0.1800)

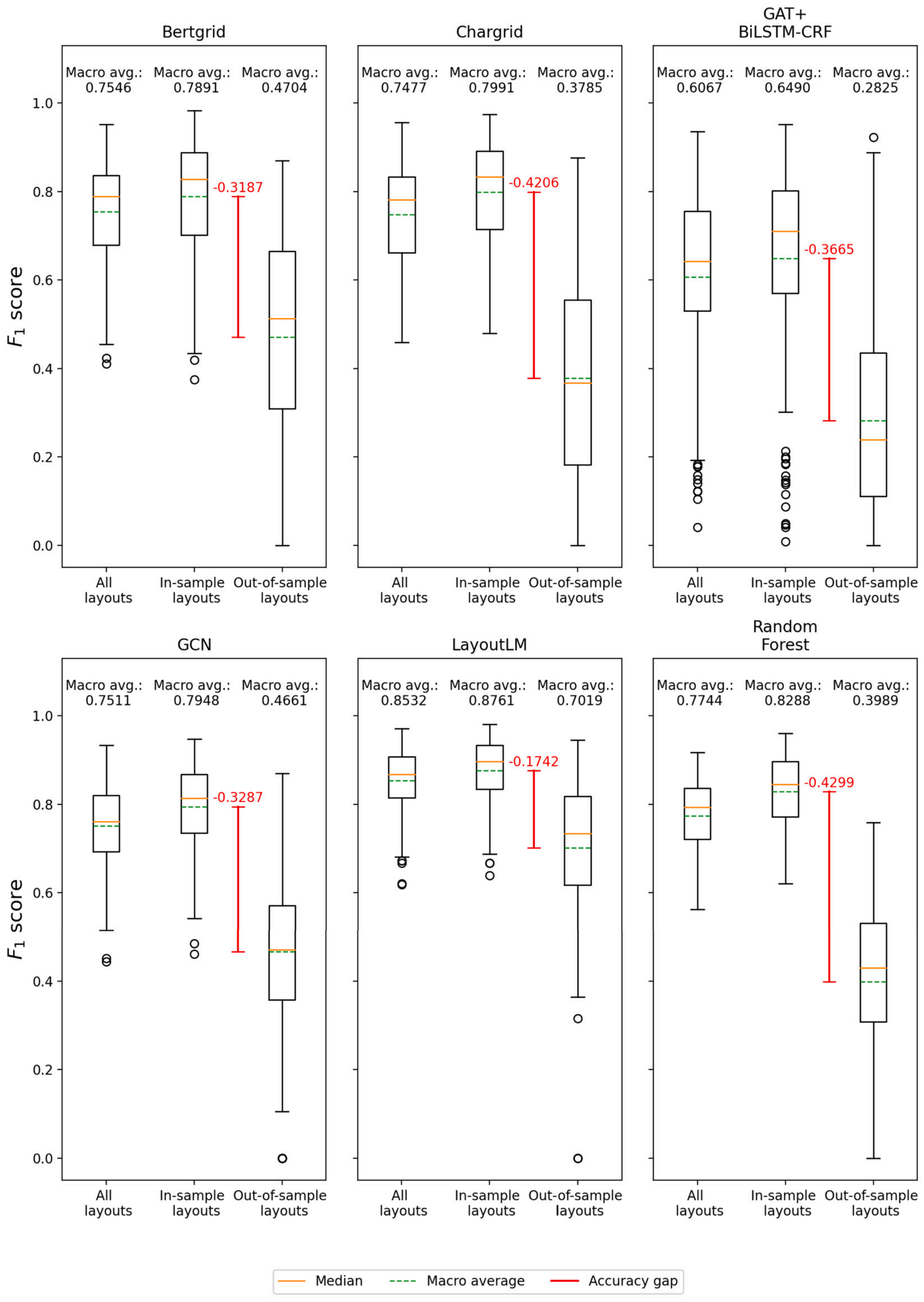

Fig. 7. Box plots of the F_1 scores across all folds for both in-sample and out-of-sample layouts by model. The red line emphasizes the gap in predictive quality between in-sample and out-of-sample layouts. The macro average is the unweighted average of class-wise F_1 scores across all folds.

Considering our aim of automating the IE from invoices from long tail suppliers, letting this bias go undetected could have had negative effects in practice as the models would yield considerable amounts of false positives and false negatives. We expected to find the existence of this bias, but we were surprised to find that the models were affected to varying degrees by this it.

Therefore, with respect to previous research, we see the main contribution of our research as discovering the layout bias and describing and implementing an evaluation methodology to detect it. The distribution of layouts has seldom been reported or methodologically addressed in the related literature. This unfortunately also holds true for the open benchmark data sets SROIE (Huang et al., 2019), CORD (Park et al., 2019) and Kleister (Stanisławek et al., 2021), due to which we suspect that the results obtained from these benchmark data sets might also be affected, albeit to a possibly different degree. A further contribution of this paper is the development and implementation of a benchmark on a dedicated invoice data set, using a common set of accuracy metrics. Previous studies of IE from layout-rich documents either relied on proprietary data sets, used different metrics and aggregation levels for evaluation, or did not report results for individual entities. Benchmarking studies dedicated to IE from invoices (Baviskar, Ahirrao, & Kotecha, 2021, Liu et al., 2016) did not include layout-aware approaches to IE.

The results further hold practical implications. They show that designers of systems that seek to automate IE from invoices should carefully evaluate the distribution of layouts in the population to which the system is applied. In the case presented in this paper, the layout bias can represent a form of selection bias (Shah et al., 2020): The invoices used for model training were taken from a ground population that contained a skewed distribution of invoice layouts. If the models are only intended to be applied to long tail suppliers, this approach will most certainly yield suboptimal outcomes, as shown in our results. If, however, the models are to be applied to the whole range of incoming invoices, failing to appropriately capture highly recurrent invoice layouts would lead to inefficiencies. In this case, a significant change in the supplier structure of the company would require a retraining of the models, and therefore also the annotation of a new set of invoices. Furthermore, the skewed distribution of layouts could be addressed using multicriterion sample weights, which take into account the distribution of layouts in addition to the distribution of classes.

Of all models in our experiment, LayoutLM shows both the best accuracy over both in-sample and out-of-sample layouts. It continues the victory march of transformer models in NLP, which benefit from extensive pretraining over large unlabeled document corpora. In this context, one surprising finding in our study was the accuracy of the random forest model. The model's overall accuracy was on a par with that of the nontransformer neural networks, and it was able to outperform the Chargrid and GAT+BiLSTM-CRF models over out-of-sample layouts. Random forests could be employed very effectively to handle cases in which all possible layouts can be represented in the training data. Another unexpected finding was how the predictive quality of the BERTgrid and Chargrid models compared to the GCN model. While Chargrid and BERTgrid are very close in overall accuracy, BERTgrid is almost 0.1 points better on the out-of-sample layouts. This hints at the fact that the semantic BERT embeddings help the model draw generalizations. The drawback of both models is the memory consumption of the respective document grids: BERTgrid especially yields large tensors with the dimensions $\{width_{target}, height_{target}, \text{features}\}$. The GCN model relies on more efficient graph representations of the shape $\{nodes, features\}$; however, it achieves results very close to those of BERTgrid. As BERTgrid and Chargrid employ object detection to detect line items, we especially expected them to perform better over line item–related entities, which was not the case. Finally, the poor performance of the GAT+BiLSTM-CRF model, both overall and on unseen layouts, might be attributed to the quality of the OCR outputs; the paragraph recognition on layout-rich documents of Tesseract is prone to errors, and the resulting paragraphs fluctuate heavily in length.

With respect to the individual entities, we observed that the models had problems extracting the total tax amount. The total tax amount appeared infrequently in our data set; only 383 text boxes have been annotated as such. The recipient address was the easiest information entity in our data set: the recipient is always the same company, and only small variations in the address appear, such as street names, house numbers, and cities for the different offices. Considering its importance to both accountants and auditors, we were disappointed to see the low performance of all models on the total amount as compared to the invoice number and issue date.

These results and the subsequent discussion have to be seen in the light of some limitations. First, the data set was rather small. The models might respond better to larger data sets and show fewer signs of bias, even if the distribution of layouts were similarly skewed. Apart from its size, the data set was also composed of mainly English invoices. Therefore, the effect of multiple languages on the predictive quality of the models was not adequately considered. Furthermore, the evaluation methodology was quite strict and did not account for further rule-based post-processing to reduce false positives and false negatives, such as checking the internal consistency of the extracted amounts. Also, most models in the experiment were our own implementations, and they might not have been true in all aspects to the original implementations. This was a necessity, as the original implementations were not openly accessible: In some cases, the authors omitted important details for the implementation, such as the ratios of the anchor boxes (Katti et al., 2018), or model parameters (Liu, Gao, et al., 2019, Lohani et al., 2019). In other cases, we did not have the respective resources; such as knowledge- and data bases (Lohani et al., 2019) or massive quantities of invoices to train a custom BERT model (Denk & Reisswig, 2019). However, we have specified all relevant details of our implementation and justifications for deviations from the original papers in section 4.

7. Conclusion

IE from invoices offers a way to automate invoice processing that causes less friction with suppliers that are not suitable for the implementation of an EDI-based invoicing tool. The results from our research show that designers of IE systems for invoices should carefully consider how to collect invoices for training and testing data, as the distribution of invoices per supplier in a company is usually biased toward a few suppliers. According to our results, this skewed distribution of suppliers, and therefore layouts, causes the models to perform worse on invoices with out-of-sample layouts, such as invoices whose layouts have not been part of the training set. However, this effect varied across the models used in our study. LayoutLM is the least affected by layout bias, while it simultaneously exhibits the best macroaveraged F_1 scores on both in-sample and out-of-sample layouts; this result can most likely be attributed to the extensive pretraining of the model. This kind of bias can go undetected if not appropriately accounted for. As it has received little attention in previous research, we also suspect that the results obtained from popular open data sets for IE from layout-rich documents might be affected. We therefore strongly encourage researchers to investigate the distribution of layouts in the respective training, validation, and test sets for these data sets.

Our study also aims to inspire future research that investigates the effects of layout distributions in the training data on model accuracy. For instance, future research could perform a more nuanced model evaluation that differentiates between e.g. zero shot, one shot, or multi shot layouts in the test data. In this context, it would also be of interest to conduct a sensitivity analysis on layout saturation, i.e., how many examples of one layout are required for the models to achieve a high accuracy on invoices of the respective layout. Such analysis may hold implications for the optimal composition of training data and model fine-tuning strategies. Besides that, future studies could explore potential means to increase the generalization capability of models in the face of skewed layout distribution. Here, we perceive synthetically created documents as one way of addressing it. The challenge hereby is to induce a sufficient layout variation in the synthetic data while still creating realistically looking documents. Another way this could be addressed is by weighting the instances of training data according to the frequency of their layouts - documents that partain to a highly occurring layout should impact the training loss relatively less than those of rarer layouts. This poses the challenge of combining the weighting of layout frequencies with the weighting of class frequencies.

CRediT authorship contribution statement

Felix Krieger: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing. **Paul Drews:** Conceptualization, Supervision, Writing – review & editing. **Burkhardt Funk:** Methodology, Supervision, Writing – review & editing.

Table 7
Information entity classes.

Key Item	Data Type	Description	Annotated text boxes
Invoice number	Alphanumeric	The unique identifier of each invoice. May appear multiple times on the same invoice.	1,147
Issue date	Date	The date on which the invoice has been issued. May appear multiple times on the same invoice.	1,377
Total amount	Decimal	The gross total amount due per invoice. Includes any tax, discounts, or other deductions.	1,096
Recipient address	Text, integer, alphanumeric	Sequence of text indicating the name, street, house number, ZIP code, and country of the invoice's recipient.	13,683
Supplier address	Text, integer, alphanumeric	Sequence of text indicating the name, street, house number, ZIP code, and country of the invoice's issuer.	12,857
Supplier VAT ID / tax ID	Alphanumeric	The (value-added) tax ID of the invoice's issuer follows a fixed pattern in EU invoices, varying across other countries, mostly not present in US invoices.	1,345
Total tax amount	Decimal or percentage	The share of the total amount due to value added or sales tax. Usually expressed as a sum or percentage of the net amount.	383
Due date	Date	The date upon which the payment is due.	822
Service date	Date	The date on which the related services have been performed or the delivery of the related products has occurred. In the case of services, this can also be a range of dates and be part of a line item.	1,292
Line item description	Text	A textual description of the goods or services provided.	13,289
Line item quantity	Integer	The quantity of goods or services provided. In the case of services, this might relate to temporal units.	1,747
Line item unit price	Decimal	The price (gross or net) per unit.	1,632
Line item tax amount	Decimal, percentage	The (value-added, sales) tax due for the goods or services provided, expressed either in an amount or as a percentage of the net price.	1,359
Line item subtotal	Decimal	The (gross or net) subtotal of each line item, equivalent to quantity x price (+ tax).	1,994

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgements

We acknowledge support by the German Research Foundation (DFG).

Appendix A. Invoice annotation

Appendix consists of Table 7.

References

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., & Vollgraf, R. (2019). FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)* (pp. 54–59).

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Gao, J., Piao, S., Zhou, M., & Hon, H.-W. (2020). UniLMv2: Pseudo-masked language models for unified language model pre-training. In *Proceedings of machine learning research: Vol. 119. Proceedings of the 37th international conference on machine learning* (pp. 642–652). http://proceedings.mlr.press/v119/bao20a.html.

Baviskar, D., Ahirrao, S., & Kotecha, K. (2021). Multi-layout unstructured invoice documents dataset: A dataset for template-free invoice processing and its evaluation using AI approaches. *IEEE Access*, *9*, 101494–101512. https://doi.org/10.1109/ACCESS.2021.3096739.

Baviskar, D., Ahirrao, S., Potdar, V., & Kotecha, K. (2021). Efficient automated processing of the unstructured documents using artificial intelligence: A systematic literature review and future directions. *IEEE Access*, *9*, 72894–72936. https://doi.org/10.1109/ACCESS.2021.3072900.

Bengio, Y. (2009). *Learning deep architectures for AI. Foundations and Trends in Machine Learning*, *2*(1), 1–127. https://doi.org/10.1561/2200000006.

Bengio, Y. (2012). *Practical recommendations for gradient-based training of deep architectures* (2nd ed.). *Neural networks: Tricks of the trade*. Springer Berlin Heidelberg (pp. 437–478).

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324.

Cristani, M., Bertolaso, A., Scannapieco, S., & Tomazzoli, C. (2018). Future paradigms of automated processing of business documents. *International Journal of Information Management*, *40*, 67–75. https://doi.org/10.1016/j.ijinfomgt.2018.01.010.

Datasnipper (2023). Datasnipper. https://www.datasnipper.com/.

Denk, T. I., & Reisswig, C. (2019). BERTgrid: Contextualized embedding for 2D document representation and understanding. arXiv:1909.04948.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186).

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch geometric. arXiv:1903.02428 [cs.LG]. https://arxiv.org/abs/1903.02428.

Forney, G. D. (1973). *The Viterbi algorithm. Proceedings of the IEEE*, *61*(3), 268–278. https://doi.org/10.1109/PROC.1973.9030.

Garncarek, L., Powalski, R., Stanislawek, T., Topolski, B., Halama, P., Turski, M., & Graliński, F. (2021). Lambert: Layout-aware language modeling for information extraction. In *Document analysis and recognition – ICDAR 2021* (pp. 532–547).

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques* (3rd ed.). *The Morgan Kaufmann series in data management systems*. Elsevier, Morgan Kauffman.

Heinzerling, B., & Strube, M. (2018). BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

Huang, Z., Chen, K., He, J., Bai, X., Karatzas, D., Lu, S., & Jawahar, C. V. (2019). ICDAR2019 competition on scanned receipt OCR and information extraction. In *2019 international conference on document analysis and recognition (ICDAR)* (pp. 1516–1520).

ISA 330, I. (2009). International standard on auditing 330 the auditor's responses to assessed risks. https://www.ifac.org/system/files/downloads/a019-2010-iaasb-handbook-isa-330.pdf.

Karnin, Z., Koren, T., & Somekh, O. (2013). Almost optimal exploration in multi-armed bandits. *Proceedings of Machine Learning Research*, *28*(3), 1238–1246. https://proceedings.mlr.press/v28/karnin13.html.

Katti, A. R., Reisswig, C., Guder, C., Brarda, S., Bickel, S., Höhne, J., & Faddoul, J. B. (2018). Chargrid: Towards understanding 2D documents. arXiv:1809.08799.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. arXiv:1412.6980 [cs.LG]. https://doi.org/10.48550/arXiv.1412.6980.

Klein, B., Agne, S., & Dengel, A. (2004). Results of a study on invoice-reading systems in Germany. In *Document analysis systems VI* (pp. 451–462).

Koch, B. (2017). Business case E-invoicing / E-billing. https://www.billentis.com/e-invoicing-businesscase.pdf.

Koch, B. (2019). The E-invoicing journey 2019-2025. https://www.billentis.com/The_einvoicing_journey_2019-2025.pdf.

Krieger, F., Drews, P., Funk, B., & Wobbe, T. (2021). Information Extraction from Invoices: A Graph Neural Network Approach for Datasets with High Layout Variety. Wirtschaftsinformatik 2021 Proceedings.

Liu, W., Zhang, Y., & Wan, B. (2016). *Unstructured document recognition on business invoice*. CS229: Machine Learning, Tech. Rep. Stanford, CA, USA: Stanford iTunes University. https://cs229.stanford.edu/proj2016/report/LiuWanZhang-UnstructuredDocumentRecognitionOnBusinessInvoice-report.pdf.

Liu, X., Gao, F., Zhang, Q., & Zhao, H. (2019). Graph convolution for multimodal information extraction from visually rich documents. arXiv:1903.11279.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692 [cs.CL]. https://doi.org/10.48550/arXiv.1907.11692.

Lohani, D., Belaïd, A., & Belaïd, Y. (2019). An invoice reading system using a graph convolutional network. In *Computer vision – ACCV 2018 workshops 11367* (pp. 144–158).

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1064–1074).

Majumder, B. P., Potti, N., Tata, S., Wendt, J. B., Zhao, Q., & Najork, M. (2020). Representation learning for information extraction from form-like documents. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 6495–6504).

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press, ISBN 978-0521865715. https://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv:1301.3781 [cs.CL]. https://doi.org/10.48550/arXiv.1301.3781.

Narayanam, K., Goel, S., Singh, A., Shrinivasan, Y., Chakraborty, S., Selvam, P., Choudhary, V., & Verma, M. (2020). Blockchain based e-invoicing platform for global trade. In *IEEE international conference on blockchain, blockchain 2020* (pp. 385–392).

Narayanam, K., Goel, S., Singh, A., Shrinivasan, Y., & Selvam, P. (2021). Blockchain based accounts payable platform for goods trade. In *IEEE international conference on blockchain and cryptocurrency* (pp. 1–5).

Palm, R. B., Winther, O., & Laws, F. (2017). CloudScan - a configuration-free invoice analysis system using recurrent neural networks. In *14th IAPR international conference on document analysis and recognition (ICDAR): Vol. 01* (pp. 406–413). arXiv:1708.07403.

Park, S. (2021). CORD: A consolidated receipt dataset for post-OCR parsing. https://github.com/clovaai/cord.

Park, S., Shin, S., Lee, B., Lee, J., Surh, J., Seo, M., & Lee, H. (2019). CORD: A consolidated receipt dataset for post-OCR parsing. Workshop on Document Intelligence at NeurIPS 2019, https://openreview.net/forum?id=SJl3z659UH.

Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. arXiv:1606.02147 [cs.CV]. https://doi.org/10.48550/arXiv.1606.02147.

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., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Probst, P., Wright, M. N., & Boulesteix, A.-L. (2019). Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery*, *9*(3). https://doi.org/10.1002/widm.1301.

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks* (pp. 45–50).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, *28*, 91–99. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Sarawagi, S. (2008). *Information extraction. Foundations and Trends in Databases*, *1*(3), 261–377. https://doi.org/10.1561/1900000003.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. arXiv:1508.07909 [cs.CL]. https://doi.org/10.48550/arXiv.1508.07909.

Shah, D. S., Schwartz, H. A., & Hovy, D. (2020). *Predictive biases in natural language processing models: A conceptual framework and overview*In *Proceedings of the 58th annual meeting of the association for computational linguistics*. Association for Computational Linguistics.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)* (pp. 464–472).

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. arXiv:1803.09820 [cs.LG]. https://doi.org/10.48550/arXiv.1803.09820.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Advances in neural information processing systems: Vol. 25*. https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

Stanisławek, T., Graliński, F., Wróblewska, A., Lipiński, D., Kaliska, A., Rosalska, P., Topolski, B., & Biecek, P. (2021). Kleister: Key information extraction datasets involving long documents with complex layouts. In *Document analysis and recognition – ICDAR 2021* (pp. 564–579).

Tanner, C., & Richter, S.-L. (2018). *Digitalizing B2B business processes—the learnings from E-invoicing. Business information systems and technology 4.0: New trends in the age of digital change*. International Publishing Springer (pp. 103–116).

Tater, T., Gantayat, N., Dechu, S., Jagirdar, H., Rawat, H., Guptha, M., Gupta, S., Strak, L., Kiran, S., & Narayanan, S. (2022). AI driven accounts payable transformation. In *Thirty-sixth AAAI conference on artificial intelligence, AAAI 2022, thirty-fourth conference on innovative applications of artificial intelligence, IAAI 2022, the twelveth symposium on educational advances in artificial intelligence, EAAI 2022 virtual event, February 22 - March 1, 2022* (pp. 12405–12413). https://ojs.aaai.org/index.php/AAAI/article/view/21506.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., Shleifer, S., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45).

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144 [cs.CL]. https://doi.org/10.48550/arXiv.1609.08144.

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1192–1200). arXiv:1912.13318.

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., Zhang, M., & Zhou, L. (2022). LayoutLMv2: Multi-model pre-training for visually-rich document understanding. arXiv:2012.14740 [cs.CL]. https://doi.org/10.48550/arXiv.2012.14740.

Yu, W., Lu, N., Qi, X., Gong, P., & Xiao, R. (2020). PICK: Processing key information extraction from documents using improved graph learning-convolutional networks. arXiv:2004.07464.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. arXiv:2106.11342 [cs.LG]. https://doi.org/10.48550/arXiv.2106.11342.

Zhang, P., Xu, Y., Cheng, Z., Pu, S., Lu, J., Qiao, L., Niu, Y., & Wu, F. (2020). TRIE: End-to-end text reading and information extraction for document understanding. In *Proceedings of the 28th ACM international conference on multimedia* (pp. 1413–1422).

Zhao, X., Niu, E., Wu, Z., & Wang, X. (2019). CUTIE: Learning to understand documents with convolutional universal text information extractor. arXiv:1903.12363 [cs]. https://doi.org/10.48550/arXiv.1903.12363.