Wirtschaftsinformatik 2021 Proceedings

Track 9: Data Science & Business Analytics

Information Extraction from Invoices: A Graph Neural Network Approach for Datasets with High Layout Variety

Felix Krieger
Leuphana Universität Lüneburg

Paul Drews
Leuphana Universität Lüneburg

Burkhardt Funk
Leuphana Universität Lüneburg

Till Wobbe
EY, GSA Assurance Research & Development

Krieger, Felix; Drews, Paul; Funk, Burkhardt; and Wobbe, Till, "Information Extraction from Invoices: A Graph Neural Network Approach for Datasets with High Layout Variety" (2021). *Wirtschaftsinformatik 2021 Proceedings*. 4.
https://aisel.aisnet.org/wi2021/RDataScience/Track09/4

Information Extraction from Invoices: A Graph Neural Network Approach for Datasets with High Layout Variety

Felix Krieger[1], Paul Drews[1], Burkhardt Funk[1], and Till Wobbe[2]

[1] Leuphana Universität, Institute of Information Systems, Lüneburg, Germany
{felix.krieger,paul.drews,burkhardt.funk}@leuphana.de
[2] EY, GSA Assurance Research & Development, Essen, Germany
till.r.wobbe@de.ey.com

Abstract. Extracting information from invoices is a highly structured, recurrent task in auditing. Automating this task would yield efficiency improvements, while simultaneously improving audit quality. The challenge for this endeavor is to account for the text layout on invoices and the high variety of layouts across different issuers. Recent research has proposed graphs to structurally represent the layout on invoices and to apply graph convolutional networks to extract the information pieces of interest. However, the effectiveness of graph-based approaches has so far been shown only on datasets with a low variety of invoice layouts. In this paper, we introduce a graph-based approach to information extraction from invoices and apply it to a dataset of invoices from multiple vendors. We show that our proposed model extracts the specified key items from a highly diverse set of invoices with a macro F_1 score of 0.8753.

Keywords: Graph attention networks, unstructured data, audit digitization, graph-based machine learning

1 Introduction

According to the study conducted by Frey & Osborne [1], auditing is among the professions which are most likely to be impacted by computerization, as they involve a high number of repetitive, structured tasks. One crucial task that fits this description is the extraction of information from invoices (EII), which is performed during tests of details. Tests of details are substantive audit procedures to obtain evidence that the balances and disclosures related to the audited company's financial statement and the corresponding transactions have been recorded and reported correctly [2]. Invoices are used frequently here, as they are the most elemental source of data used in accounting. They hold the details of any commercial exchange of goods or services between companies and/or consumers. Details of interest to the auditor are e.g. invoice numbers, invoice and due dates, total and tax amounts, VAT numbers, and line items. When performing tests of details, auditors draw samples of invoices, which can range from dozens to hundreds of documents in size, depending on the beforehand conducted risk

16th International Conference on Wirtschaftsinformatik,
March 2021, Essen, Germany

assessment. Reviewing the sampled invoices by hand for tests of details requires many person-hours per audit engagement. Automating EII can thus increase the efficiency of audits, while simultaneously increasing audit quality by allowing auditors to focus on higher value-added tasks, and through the ability to test more invoices by increasing the processing speed. Initially proposed solutions for automating EII employ rules-based processing and template-matching [3–5], which require human input to construct business rules and templates. In an audit context, the scalability of such solutions quickly reaches its limits: Especially bigger audit firms audit a wide range of clients from multiple industries, which receive invoices from a multitude of different business partners. The layouts of invoices can vary highly between issuing companies (hereafter referred to as 'vendors'). The efficiency gains from rules- or template-based automation solutions would soon be canceled out by the effort required for their adaption to individual vendor layouts. For a solution to be employed in audits, it should therefore be able to capture the general patterns prevalent on invoices and generalize to unseen invoice layouts. The applicability of such a solution would also extend beyond auditing and could support administrative processes, especially accounts payable, in public and private organizations. To address the complexity of dealing with a multitude of invoice layouts, recent research in the area of EII has proposed machine learning (ML)-based approaches [6–11]. The challenge for the application of ML to EII is that the text follows a 2-dimensional layout, as opposed to the sequential, unformatted text usually assumed by natural language processing (NLP) methods. Previous studies proposed to employ graphs for representing the text on an invoice document such that the layout is preserved [9, 10]. The key items are extracted from this document graph by using graph convolutional neural networks (GCN) [9, 10]. GCN leverage a context diffusion mechanism, which is functionally similar to the local receptive fields in (grid) convolutional networks used in computer vision (CV) [12]. Graph representations of invoices are albeit less granular than their pixel-based CV counterparts [7, 8], making them more computationally efficient. However, the ability to extract key items from invoices of GCNs has so far only been demonstrated on invoice datasets with minor variations in layouts [10]. In line with the above-mentioned requirements for the audit domain, our research therefore addresses the following research question:

"*How can graph-based neural networks be applied to extract key items from invoices with a high variety of layouts?*"

The contribution of this paper lies in the introduction of a graph attention-based model to extract information from invoices, and its application on a dataset of invoices sourced from a multitude (277) of vendors.

2 Related Work

Early works concerned with the automation of EII have studied rule- and template-based approaches to automating this task [3–5]. These approaches are able to extract the desired information, albeit only from known invoice templates, and require human input to create new rules or templates. One of the first proposed systems to apply ML was CloudScan [6], which uses an LSTM-based neural network to extract key items via sequence labeling, a common approach to information extraction in NLP. However, such approaches assume the text to be sequential and unformatted and do not account for the 2-dimensional layout of invoices.

Recent studies have proposed different approaches to represent the text on invoices such that the layout is preserved. The approaches can be broadly classified into grid-based [7, 8] and graph-based [9, 10]. In the former, the text is mapped to a grid, as in *Chargrid* [7] and *BERTgrid* [8]. The latter approaches model documents as graphs, in which either words [9] or whole text segments [10] are represented as nodes, and their spatial relationships are represented as edges. In [10], the edges are furthermore weighted with the distances between text segments. As is shown in table 1, the document representation and the methods used are intertwined. Table 1 summarizes the methodology of previous studies and provides details on the respective data sets and the reported performance metrics: Grid-based approaches lean methodologically more towards CV and define the task of extracting key items as semantic segmentation and/or bounding box regression [7, 8]. The graph-based approaches lean more towards NLP, using word/node classification and sequence labeling to identify key items. [9, 10]. Majumder et al. [11] use a different approach. Their work is based on representation learning and leverages prior knowledge about key items which is used to generate prototypical embeddings for each key item. For each field, candidate words are selected based on their inferred data type. To determine whether a candidate is a key item, it is embedded together with its contextual information, and the cosine distance between the candidate's embedding and the respective key item embedding is calculated [11].

While there are different approaches to document representation, a notion common to all mentioned works is the importance of context for the detection of key items. To this end, grid [7, 8] or graph [9, 10] convolutions are employed in the recent literature, as well as the attention mechanism [11], or a combination thereof [10]. Further similarities can be found in the nature of the features used. Usually, some combination of syntactical, positional, and/or semantic features are employed. Syntactical features capture (dis-) similarities in the syntax of words and are obtained via character [7, 8] or byte pair [9] encoding, as well as through the inference of data types (string, date, alphanumeric, etc.) [9, 11]. Positional features are usually bounding box coordinates either used as explicit features [11], implicitly encoded in the document representation [7–10], or Euclidean distances between text boxes [9, 10]. Semantic features are obtained through word embedding layers [11] or from language models such as word2vec [10] or BERT [8]. In addition to semantic, positional, and semantic features, Lohani et al. [9] use external databases to discover known entities (cities, zip codes, etc.). While the general type of features used is similar across approaches, the specific utilization and the respective implementation of the model vary. The works reviewed in this section are difficult to compare in terms of performance, as each work relies on different proprietary invoice datasets. As is shown in table 1, the datasets vary in size and variety; Majumder et al. [11] use the most exhaustive dataset of the works presented in this section, with each invoice coming from a different vendor. The dataset used by Katti et al. [7] and Denk et al. [8] is comparable in size and variety. Liu et al. [10] use a set of Chinese invoices, which all follow the same government-regulated layout. In terms of size, it is comparable to the dataset used by Lohani et al. [9]. The authors however do not provide any further details, such as the number of vendors or templates or the exact distribution of languages.

Table 1. Methodology, dataset details and reported performance measures of related studies

	Document representation structure and granularity	Model type	Information extraction task	Dataset size	Dataset languages	Number of vendors / layouts in dataset	Reported averaged performance over all key items
Denk et al. [7]	Grid; Characters	(Grid) Convolutional Neural Network	Semantic segmentation, bounding box regression	12.000	Several, mainly English	Most vendors appear once or twice [7]	61.48% Accuracy measure (as reported in [8])
Katti et al. [8]							65.49% Accuracy measure
Liu et al. [10]	Graph; Text segments	Graph Attention Network, BiLSTM-CRF	Sequence labelling	3.000	Chinese	Single layout	0.881 F_1 score
Lohani et al. [9]	Graph; Words	Graph Convolutional Network	Node classification	3.100	English, French	No reference	0.93 F_1 score (Micro) 0.93 Precision (Micro) 0.929 Recall (Micro)
Majumder et al. [11]	Candidate representations; Words	Attention-based Neural Network	Measuring candidate embedding similarity to field embedding	14.327	English	14,327	0.878 F_1 score (Macro)

Apart from the datasets, another difficulty in comparing approaches is given through the different evaluation methodologies and metrics. Katti et al. [7] and Denk et al. [8] evaluate the performance of their models on the character level. To this end, they use a metric similar to the word error rate. As they are based on the same dataset and use the same metric, they are the most comparable works reviewed in this section. Denk et al. [8] show that *BERTgrid* is able to outperform *Chargrid* with 65.49% average accuracy over 61.48%. by extracting BERT features for every word on the invoice from a BERT model trained on invoices. The other works evaluate their models on the word level. Lohani et al. [9] present very detailed results for the most exhaustive list of extracted key items of all works reviewed in this section. They report F_1 scores, precision, and recall for 27 extracted key items, with micro-averages of 0.93, 0.93 and 0.929 respectively. The other graph-based approach presented by Liu et al. [10] achieves an averaged F_1 score of 0.881 on an invoice dataset. No details on the averaging method are provided. They furthermore report F_1 scores for 6 out of 16 extracted key items on the invoice dataset. Majumder et al. [11] present F_1 scores for 7 extracted key items along with a macro-averaged F_1 score of 0.878. Naively observed, it may seem that the approach introduced by Lohani et al. [9] performs better than the approaches presented by Liu et al. [10] and especially Majumder et al. [11]. Due to the specifics of the datasets, a direct comparison of the presented results is not meaningful. The limited variety of the invoice dataset used by Liu et al. [10] and the incomplete information regarding the variety of the dataset used by Lohani et al. [9] leave doubt as to whether graph-based approaches would perform as well in a setting with a higher variety of layouts. Our work aims to close this research gap by exploring the performance of a graph-based neural network for EII on a dataset with invoices from a multitude of vendors.

3 Methodology

To evaluate the performance of graph-based models in EII, we introduce a graph network model that draws inspiration from the above presented recent research. Figure 1 depicts the document representation and model architecture, and how they are intertwined. The model takes document graphs as input, in which each node represents a word in the document. Syntactic, positional, and semantic features are attached to each node, which are derived from the word the node represents. The edges in the document graphs represent the relative positional relationship between the words. Key items are then extracted via node classification. In this section, we introduce the representation of documents through node features and document graphs and the architecture of the proposed model.

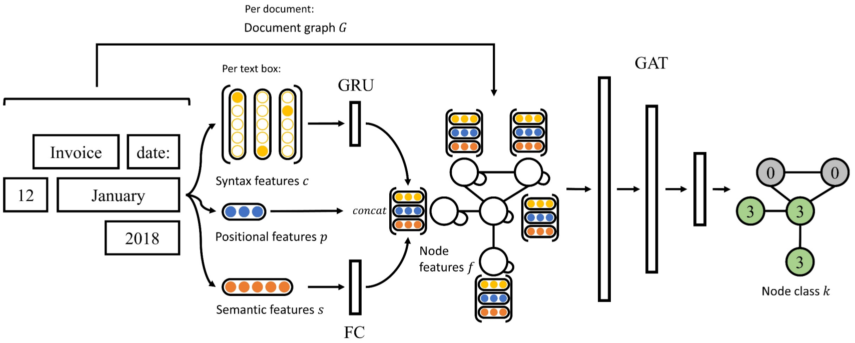

Figure 1. The node features are embedded using fully connected (FC) and recurrent (GRU) layers and are attached to the document graph, which is passed into graph attention layers (GAT) for node classification

3.1 Document representation

As mentioned before, we use graphs to represent documents, which are constructed from optical character recognition (OCR) outputs of invoices. We use word-level OCR outputs, such that OCR yields a set of text boxes $\mathcal{D}$, which contains all n recognized words w on the document. Each text box corresponds to a word. We ignore empty and whitespace text boxes. The bounds of the text boxes are described by the cartesian coordinates x_1 , y_1 , x_2 , y_2 of the box' corners, measured in pixels. The width and height of a document are denoted by W, H such that $0 \leq x_{1,2} \leq W$ and $0 \leq y_{1,2} \leq H$. An OCRed document can hence be formally described as a set of n text boxes on a 2-dimensional plane $\mathcal{D} = \left\{(w^{(j)}, x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}) \mid j \in \{1, \dots, n\}\right\}$, where the superscript refers to a text box. Each text box in $\mathcal{D}$ is represented as node in the document graph $G = (V, E)$. $V = \left\{v^{(i)} \mid i \in \{1, \dots, n\}\right\}$ is a set of nodes, and $E = \left\{e_{ij} \mid i, j \in \{1, \dots, n\}\right\}$ a set of edges between nodes $v^{(i)}$ and $v^{(j)}$. Figure 2 depicts an example of the graph representation used in our approach. E is then constructed from the text box coordinates. We use the following algorithm to construct E: Using the bounding box coordinates, each node $v^{(i)}$ is connected through e_{ji} with its neighbors $v^{(j)}$ to the top, bottom, left and right. The neighborhood $\mathcal{N}(i)$ of $v^{(i)}$ are all nodes which are connected to it via an edge: $\mathcal{N}(i) = \{v^{(j)} \in V \mid e_{ji} \in E\}$. $\mathcal{N}(i)$ can contain more than four elements, as the edges are in rare instances not symmetrical.

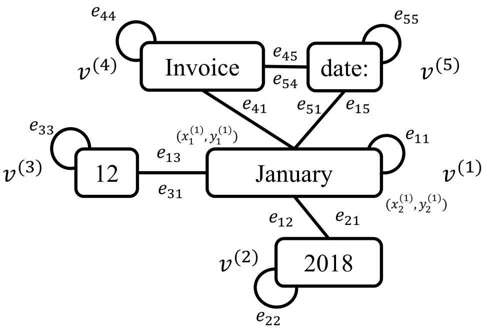

Figure 2. Example for the constructed document graph, showing the neighborhood $\mathcal{N}(1)$ for the node $v^{(1)}$ representing $w^{(1)}$ "January"

For $v^{(j)} | j \neq i$ to become a candidate for a horizontal neighbor, it must fulfill either $x_2^{(j)} \leq x_1^{(i)}$ (left neighbor) or $x_2^{(j)} \geq x_1^{(i)}$ (right neighbor), while simultaneously fulfilling $y_1^{(i)} \leq y_1^{(j)} \leq y_2^{(i)}$, $y_1^{(i)} \leq y_2^{(j)} \leq y_2^{(i)}$ or $y_1^{(j)} \leq y_1^{(i)}, y_2^{(i)} \leq y_2^{(j)}$. From these candidates, the candidate with the smallest Euclidean distance between the respective outer coordinates is then selected as neighbor. An example for this heuristic is given in Figure 3.

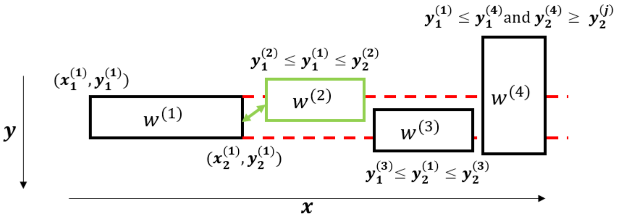
$y_1^{(1)} \leq y_1^{(4)}$ and $y_2^{(4)} \geq y_2^{(j)}$
$y_1^{(2)} \leq y_1^{(1)} \leq y_2^{(2)}$
$(x_1^{(1)}, y_1^{(1)})$
$w^{(1)}$
$w^{(2)}$
$w^{(3)}$
$w^{(4)}$
y
$(x_2^{(1)}, y_2^{(1)})$
$y_1^{(3)} \leq y_2^{(1)} \leq y_2^{(3)}$
x

Figure 3. Valid right neighbor candidates for $v^{(1)}$, with $v^{(2)}$ being selected as neighbor

Vertical neighbors are determined analogically. In addition to the neighbors, each node includes a self-loop e_{ii}. Through the self-loops, the model proposed in section 3.2 can access the node's own features. The edges in E are unweighted and undirected, the in-degree of $v^{(i)}$ is equal to its out-degree.

For each node in G, word-level features are extracted. We use three types thereof: Syntactic features, positional features, and semantic features. The syntactic features are used to capture the fine-grained syntactical (dis-) similarities between tokens. Character level one-hot representations of $w^{(j)}$ are extracted, using a fixed dictionary of 110 capital and lowercase Western European letters, numbers and special characters. The length of each token is padded to ten characters. The one-hot encoding process yields a tensor $\mathbf{C} = [\boldsymbol{c_1}, \ldots, \boldsymbol{c_n}]$ with dimensions $(n \times 110 \times 10)$. The coordinates of the text boxes are used to extract positional features. The $x_{1,2}^{(j)}$ *and* $y_{1,2}^{(j)}$ coordinates are scaled to W and H respectively. The positional features include the width, height, and area of the text box, and the Euclidean distances to the nearest neighbors. Missing values for distances are imputed using the maximum possible distance 1.0. Missing values appear if a node has no neighbor in one of the directions. In sum, 13 positional features are extracted for each text box, yielding a matrix $\mathbf{P} = [\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}]$ of shape $(n \times 13)$. The semantic features are supposed to capture the meaning behind a token and its relationship to other tokens. To this end, we extract word embedding vectors for $w^{(j)}$ using a pretrained multilingual BERT [13] base model. To ensure the scalability of the system, we refrain from building dictionaries or embedding tables of a fixed set of words. Another reason for this is the susceptibility of OCR to noise, depending on the quality of the underlying document. For each $w^{(j)}$, BERT outputs a feature vector of length 768. Passing each token into BERT, a feature matrix $\mathbf{S} = [\boldsymbol{s_1}, \ldots, \boldsymbol{s_n}]$ is obtained. Four model inputs are generated in summary: The document graph *G*, a feature tensor containing the one-hot encoded tokens **C,** the positional feature matrix **P**, and the semantic feature matrix **S**.

3.2 Model architecture

We use the document representation described above as input to the model to perform node classification on the document graph *G*; each node is assigned a corresponding class label. The model proposed in this paper is composed of recurrent, linear, and graph attention layers. Figure 1 shows the model architecture, along with the correspondent in- and outputs. The first layers are designated for feature embedding. The one-hot encoded character sequences **C** are passed into a gated recurrent unit (GRU) [14] layer, which extracts syntax-sensitive word embeddings **C'**. Recurrent layers (such as GRU) have been shown to be useful for character-level language modeling [15]. The semantic features **S** are embedded into a lower dimensional vector space using a fully connected layer (FC) with $ReLu$ nonlinearity, yielding **S'**. The embedded syntactic and semantic features are then concatenated with the positional features to form the node features $\mathbf{F} = (\mathbf{C}' \,||\, \mathbf{P} \,||\, \mathbf{S}')$, where $||$ denotes the concatenation operator. *G* and **F** are passed into the graph attention (GAT) layers [16]. GAT layers perform weighted neighborhood feature aggregation over *G*, using attention scores as weights. The attention mechanism allows the model to focus on specific neighboring nodes. For a node $v^{(i)}$ and its neighborhood $\mathcal{N}(i)$ of adjacent nodes, which includes its self-loop e_{ii}, the forward propagation rule of a GAT layer l can be written as

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\right) \tag{1}$$

where $\alpha_{ij}^{(l)} = softmax\left(w_{ij}^{(l)}\right)$ and $w_{ij}^{(l)}$ are the raw attention scores $w_{ij}^{(l)} = LeakyRelu\left(a^{(l)^T}\left(z_i^{(l)} \| z_j^{(l)}\right)\right)$. $a^{(l)^T}$is a vector learned by the model and $z_i^{(l)}, z_j^{(l)}$ are the linear activations of layer l. σ denotes the nonlinearity, for which we use $ReLu$ on the GAT layers, similar to Lohani et al. [9]. We use multiple GAT layers to extend the context used to classify a node beyond its direct neighborhood $\mathcal{N}(i)$ to include the neighborhoods $\mathcal{N}(j)$ of the nodes in $\mathcal{N}(i)$ [12]. In the first two GAT layers, we additionally employ multi-headed attention. Multi-headed attention has been applied to increase the stability of the learning process in GAT layers [16]. The attention heads compute equation (1) independently, their outputs are then concatenated and passed into the next layer. The last layer is a single-headed GAT layer with a $softmax$ activation function, which performs the node classification. It returns a probability distribution over the classes for each node in the document graph**.**

4 Experimental setup

We test the above-proposed document representation and model using a set of invoices to determine its performance in a realistic setting. For now, we focus on a limited set of key items to be extracted: The *invoice number*, the *invoice date*, and the *total amount*. We define a fourth class, "*unlabeled*", for all other text on the invoice. Details on the dataset and the model implementation and training are given below.

4.1 Dataset

As of now, there are no publicly available sets of labeled invoices, which are sufficient in size and variety to train sophisticated ML models. For this research, we were provided a set of invoices by an audit firm, in which they are the recipient. The dataset is composed of 1129 English one-page invoices from 277 different vendors. We annotated the invoices ourselves by hand for the key items. Individual key items can be composed of multiple words and appear more than once on one invoice. The classes (i.e. key items) in our dataset are sharply imbalanced: Out of the 243,704 textboxes retrieved in our dataset, only 1427 (~0.58%) contain *invoice numbers,* 2221 (~0.91%) contain *total amounts,* and 2600 (~1.06%) contain *invoice dates*. Table 2 details the split of the dataset into training, validation, and test sets. We chose validation and test set sizes of 10% each, to save as many examples for training as possible. The data splits were stratified across vendors. This way we ensure that both the validation and test splits contain invoices from vendors that remained unseen during training.

Table 2. Details on data splits

	Number of invoices	Number of unique vendors	Number of vendors unique to split
Training set	903	239	178
Validation set	113	58	18
Test set	113	62	18

4.2 Implementation and training

The model described above is implemented using Pytorch 1.7.0 with CUDA 10.1 and the Deep Graph Library [17] 0.5.2. We use Tesseract 4.0 as OCR engine. Table 3 outlines the chosen model specification in terms of layer sizes (number of hidden nodes) and the number of attention heads in the GAT layers. The size of the GAT 1 layer equals the sum of sizes of the FC and GRU layers and the number of positional features $\boldsymbol{p}$.

Table 3. Selected model specification

Layer	Size	# Attention heads
FC	256	-
GRU	128	-
GAT 1	397	12
GAT 2	192	8
GAT 3	512	1

The model is trained using the multi-class cross-entropy loss between the predicted and target labels for the nodes, with class weighting to address the class imbalance described above. The ‘unlabeled’ class is weighted with 0.1112 and the key item classes with 1.0. The weighting increases the misclassification cost for key items compared to the ‘unlabeled’ class. Training batches are constructed from batches of documents, using 8 invoices per batch. The invoices in the training set are shuffled. ADAM [18] is used as optimizer with the standard configuration $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e - 8$. We employ gradient clipping to control for the exploding gradients problem in recurrent layers; all gradients with L^2 norm over 0.5 are clipped. A fixed stepwise learning rate (α) schedule is applied: The model training starts with $\alpha = 5.4452 * 10^{-4}$, and α is decreased by factor 10 every 50 epochs. We furthermore use an early stopping criterion, which aims to maximize the macro F_1 score on the validation set with a patience of 50 epochs.

The above described model specification (layer sizes, number of attention heads per layer) and training hyperparameters (α, batch size, weighting of the “unlabeled” class) were selected using a hyperparameter search with Hyperband [19]. 50 hyperparameter configurations were tried with the objective to maximize the macro F_1 score on the validation set. For each configuration, the model was trained for a minimum of 10 and a maximum of 60 epochs. The best configuration achieved a macro F_1 score (incl. the “unlabeled” class) of 0.8956 after 60 epochs.

5 Results

We report F_1, precision, and recall scores for the extracted key items. We furthermore include their macro averages, both including and excluding the unlabeled class. The scores are calculated by comparing the model outputs with the annotated instances in the test set. The model outputs were generated using the model state after the completion of epoch 64, as the early stopping criterion ended the model training on epoch 115. We furthermore analyze the attention weights inferred on the document graph edges by the model.

5.1 Classification results

Table 4 shows the classification results on the test set. The model is able to detect all three key items, though performing better on *invoice numbers* and *invoice dates* than *total amounts*. For *total amounts* and *invoice dates*, precision and recall are well balanced, leading to reasonable high F_1 scores. For *invoice numbers*, the spread between precision and recall is higher.

Table 4. Classification results on test set

	Unlabeled	Invoice number	Total amount	Invoice date	Macro avg. incl. unlabeled	Macro avg. excl. unlabeled
Precision	0.9959	0.9391	0.8333	0.9196	0.9220	0.8974
Recall	0.9971	0.8571	0.8072	0.8996	0.8902	0.8546
F_1 Score	0.9965	0.8963	0.8200	0.9095	0.9055	0.8753

This can also be seen in Figure 4, which shows a plot of the precision and recall scores for different probability thresholds (precision-recall curve), along with isometric lines for several levels of F_1 scores. The curve for *invoice numbers* indicates high trade-off costs between precision and recall if a recall over 0.9 was to be achieved. Generally, the figure shows that higher F_1 scores are attainable through threshold moving; F_1 scores over 0.9 for *invoice numbers* and close to 0.85 for *total amounts* could be achieved.

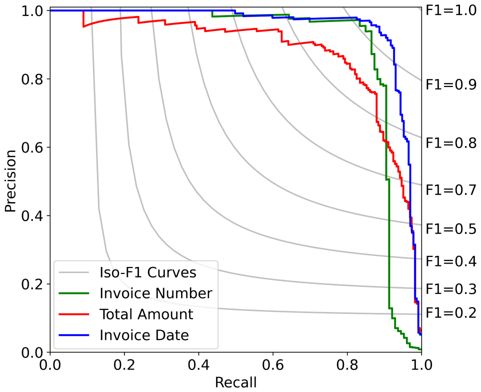

Figure 4. Precision-recall curves for different probability thresholds for the key items *invoice number*, *total amount,* and *invoice date*

5.2 Attention analysis

To classify a node $v^{(i)}$, the model has not only access to the node's own features but to all features of $\mathcal{N}(i)$. The distinguishing ability of GAT is to perform feature aggregation weighted by the attention weights allocated to the connecting edges e_{ij}. Analyzing the attention weights inferred by the model therefore allows to gain an understanding if and how contextual relationships affect the node classification. To this end, we analyze the attention weights allocated by the model on the edges of the document graphs in the test set.

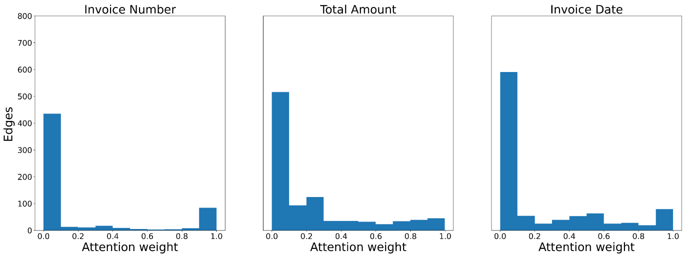

Figure 5. Distribution of attention weights over all edges on predicted key item nodes

Figure 5 depicts the distributions of attention weights inferred by the last GAT layer on the edges which connect all nodes classified as key items with their surrounding nodes. For *invoice numbers*, the model infers sharp attention weights, i.e. the weights tend to be closer to either 0.0 or 1.0, resembling a bimodal distribution. For *invoice dates* and *total amounts,* this distribution is flatter; weights in between the two extremes are allocated more frequently. As the model allocates weights close to 0.0 very often, we furthermore narrow the analysis down to the attention allocated towards the self-loops. This way, we can analyze whether the model disregards the neighborhood features in favor of the node features or vice versa.

Figure 6 shows the distribution of attention weights on the node's self-loop edges e_{ii}. The model infers primarily very small attention weights for *invoice numbers*, which indicates that the classified node's own features informed the classification not as much as the features of its neighboring nodes $v^{(j)}$. In the case of *total amounts* and *invoice dates*, the self-loop edges received much higher attention weights. In summary, the attention weights on the self-loops imply that *invoice numbers* are rather identified via their context, whereas *total amounts* and *invoice dates* are identified via their own features.

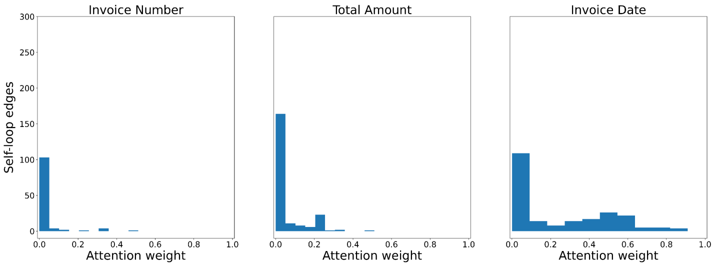

Figure 6. Distribution of attention weights over self-loop edges on predicted key item nodes

This is also reflected in Figure 7: The figure depicts graphs that summarize the attention weights inferred on the edges of the 2-hop neighborhood of predicted key items. The attention weights are summed by similar words. The thickness of the edges connecting the tokens reflects the attention placed by the model on the respective relationships, summed across all document graphs in the test set. For each key item, we exemplarily choose the 10 terms which have received the highest attention weights. In the figure, <Key Item> denotes a neighboring token which has also been classified as a key item, <Self> denotes the attention on the self-loop.

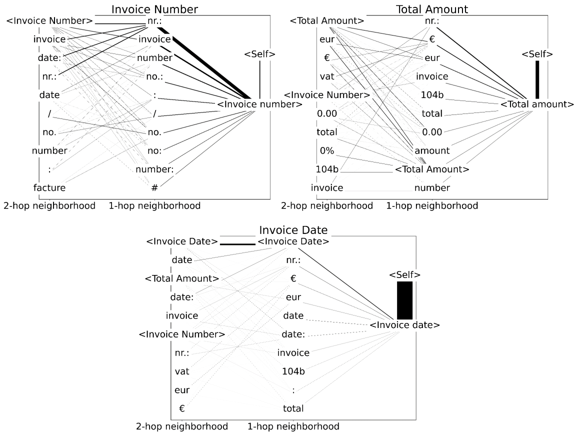

Figure 7. Attention weights allocated towards the edges of extracted key items, summed across unique words

In the case of *invoice numbers*, the model assigns most attention to combinations of words that form some variation of "invoice number." For the *invoice date*, the model allocates most attention to the self-loop and to neighboring nodes which have also been classified as invoice dates. Similarly, the classification of *total amounts* is mainly based on the node's own features, and the context receives only small attention weights. For these two key items, their context is less important for their classification than their own features.

The analysis of attention weights shows that the model classifies nodes both based on the node's own features, as well as based on the features of neighboring nodes. The weighting hereby varies by key item class. Furthermore, it shows that the model is capable of identifying contextual relationships over multiple hops in the graph. This is highly relevant as discriminative terms such as "invoice number", "total amount" and "invoice date" are usually composed of multiple words.

6 Discussion

Prior research has presented promising results for graph-based approaches to EII. However, these were tested on invoice datasets with a low variety of invoice layouts. The purpose of this research is therefore to evaluate the effectiveness of a graph-based approach on a dataset containing invoices from a multitude of vendors, based on a novel model architecture. Our results show that the model is able to capture the patterns prevalent on invoices to extract the defined key items. They further show that the model emphasizes either the context of a word or the word itself to classify it, depending on the key item class. One interesting finding is that the model identifies relevant context over multiple hops in the graph, thereby combining multiple words.

The intended area of application for the model proposed in this paper is the test of details in audits. The overall goal of the test of details is to see whether the details (key items) from the invoices have been recorded correctly. To this end, the key items extracted by the model are reconciled against other, mostly structured, sources of data. Hereby, false positives and false negatives incur different costs with respect to the overall goal of automating EII: False positives (falsely detected key items) require the active attention of an auditor. False negatives (falsely not detected key items), can be offset by further testing of details until sufficient audit evidence is collected. In that regard, the current performance of the model is adequate, yet offers room for further improvement. The model achieves macro averaged precision and recall of 0.8974 and 0.8546 on the key items. While the current performance of the model would not be enough to fully automate this task, it can still lead to efficiency gains in an audit engagement, as long as the effort to review possible false positives is smaller than the effort to extract all key items from the invoices by hand. The raw outputs of the model could also be further enhanced by heuristics and business rules to reduce false positives, which we did not explore in this paper. For example, rules could be applied which retrieve only the key items with the highest probability per document. Another possibility is to perform logical checks on the data type of a retrieved key item, e.g. evaluating whether retrieved *invoice dates* can be parsed as dates. For full automation of the task, the model should be tunable such that precisions close to 1.0 can be achieved by threshold moving, without sacrificing too much recall. In that case, it could substitute human labor by only requiring limited amounts of additional testing.

In the usage of graphs, our work relates to the approaches by Lohani et al. [9] and Liu et al. [10]. As anticipated, we do not match their respective results. We achieve F_1 scores on the *invoice number*, *total amount*, and *invoice date* of 0.8963, 0.8200, and 0.9095, while Liu et al. [10] report 0.961, 0.910, and 0.963 for the respective key items, and Lohani et al. [9] report 0.90, 0.99, and 0.95. For Liu et al. [10], this gap in performance can be attributed to the much higher layout variety in our dataset. As Lohani et al. [9] do not report in detail on the variety in their dataset, we can only safely attribute this performance gap to the difference in size: Our dataset is smaller than the invoice datasets used in related works. The differences in variety and size render the results hardly comparable. This also applies to a comparison with the results presented by Majumder et al. [11], which use the biggest and most diverse dataset in all of the related research. However, similar to our results and the results of Liu et al. [10], they report worse performance on *total amounts* than the other two key items. They achieve F_1 scores of 0.949 and 0.940 on *invoice numbers* and *invoice dates*, and 0.858 on *total amounts*. A possible explanation for this is that *total amounts* can appear multiple times on one invoice and are difficult to identify by their context.

In general, we see advantages to the graph approach: The model can access an arbitrary number of direct or indirect neighbors of each node, instead of restricting it to a fixed-sized number of neighbors, like in the approach used by Majumder et al. [11]. This advantage is however contrasted by the computational cost of constructing the document graph in the first place. To extract node embeddings from the document graph, we use GAT layers, similar to Liu et al. [10]. This way, edges between nodes can be individually weighted, unlike the graph convolutions used in [9]. GAT networks are also better suited for transductive graph learning tasks [16]. Hence, document graphs can be processed individually instead of requiring one large graph composed of multiple documents for both learning and inference. Our approach deviates from Liu et al. [10] in the granularity of the graph: Similar to [9], we construct the graph from single words instead of using paragraph-like text blocks.

The presented research must be seen in light of some limitations, which are mainly grounded in the dataset. First, multiple recurring vendors are present in the dataset, hence recurring layouts. Though we try to control for this effect by applying stratified sampling in the training, validation, and test splits, overfitting might still be an issue. Furthermore, all invoices in our dataset are addressed to the same recipient. These however are realistic circumstances for the audit domain, where a client might have recurrent business with suppliers, and all invoices are addressed to the client. Second, we only used English invoices. We have yet to assess how the model responds to invoices from several languages. A further limitation is grounded in the size of the dataset; compared to other works in the area, our dataset is quite small. The classification results of our models are therefore not comparable to other research.

7 Conclusion and Outlook

EII is a highly structured, repetitive task in auditing, which can highly benefit from automation. The high variety of invoice layouts faced by auditors calls for approaches that are able to capture the general patterns on invoices. Recent research in this area has proposed graph-based approaches to EII, showing promising results. However, these approaches have been so far applied to datasets with a low variety of invoice layouts. In this paper, we introduce a novel graph-based model architecture, perform an experiment using a dataset from 277 different vendors. The dataset resembles a realistic setting faced by auditors. We show that the model extracts specified key items with high F_1 scores, by leveraging contextual relationships between words on the invoices. While our results do not match the scores achieved by previous works due to the higher variety of invoice layouts in our dataset, they indicate that graph-based models are capable of learning the general patterns prevalent on invoices and extrapolate them.

As of now, we do not see our research on this topic as concluded. As the dataset used in this research is small compared to other related works, further research with bigger datasets and more invoice layouts needs to be conducted to strengthen our results. Complementary to that, we aim to explore more architectural options for the model, such as replacing the GRU with a convolutional layer to extract character-level word embeddings. In our usage of graphs, we further aim to explore edge features. Interesting features could be both continuous, such as the semantic and spatial distance between words, as well as categorical, such as whether words are linked entities or the direction of the edge. Further research should also include more key items to be extracted; especially line items represent an interesting area of investigation. We also plan to extend this model to extract key items from further document types such as receipts, purchase orders, etc.

As pointed out in section 2, the whole research field of EII lacks comparability. Unfortunately, as of now, there are no publicly available labeled sets of invoices that are sufficient in size and variety to train sophisticated ML models. It could therefore vastly benefit from a study that benchmarks different approaches on the same dataset, or from an openly accessible annotated dataset.

References

1. Frey, C.B., Osborne, M.A.: The future of employment: How susceptible are jobs to computerisation? Technological Forecasting and Social Change. 114, 254–280 (2017). https://doi.org/10.1016/j.techfore.2016.08.019.
2. IAASB: International Standard On Auditing 330: The Auditor's Responses To Assessed Risks (2009). https://www.ifac.org/system/files/downloads/a019-2010-iaasb-handbook-isa-330.pdf.
3. Esser, D., Schuster, D., Muthmann, K., Berger, M., Schill, A.: Automatic indexing of scanned documents: a layout-based approach. Doc. Recognit. Retr. XIX. 8297, 82970H (2012). https://doi.org/10.1117/12.908542.
4. Dengel, A., Bertin, K.: smartFIX: A Requirements-Driven System for Document Analysis and Understanding. Lect. Notes Comput. Sci. 2423, 272–282 (2002). https://doi.org/10.1007/3-540-45869-7.
5. Schuster, D., Muthmann, K., Esser, D., Schill, A., Berger, M., Weidling, C., Aliyev, K., Hofmeier, A.: Intellix – End-User Trained Information Extraction for Document Archiving. In: 2013 12th International Conference on Document Analysis and Recognition. pp. 101–105 (2013). https://doi.org/10.1109/ICDAR.2013.28.
6. Palm, R.B., Winther, O., Laws, F.: CloudScan - A Configuration-Free Invoice Analysis System Using Recurrent Neural Networks. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). pp. 406–413. IEEE, Kyoto (2017). https://doi.org/10.1109/ICDAR.2017.74.

7. Katti, A.R., Reisswig, C., Guder, C., Brarda, S., Bickel, S., Höhne, J., Faddoul, J.B.: Chargrid: Towards Understanding 2D Documents. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 4459–4469. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/D18-1476.
8. Denk, T.I., Reisswig, C.: BERTgrid: Contextualized Embedding for 2D Document Representation and Understanding. arXiv:1909.04948 [cs]. (2019).
9. Lohani, D., Belaïd, A., Belaïd, Y.: An Invoice Reading System Using a Graph Convolutional Network. In: Carneiro, G. and You, S. (eds.) Computer Vision - ACCV 2018 Workshops. pp. 144–158. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-21074-8_12.
10. Liu, X., Gao, F., Zhang, Q., Zhao, H.: Graph Convolution for Multimodal Information Extraction from Visually Rich Documents. In: Proceedings of the 2019 Conference of the North. pp. 32–39. Association for Computational Linguistics, Minneapolis - Minnesota (2019). https://doi.org/10.18653/v1/N19-2005.
11. Majumder, B.P., Potti, N., Tata, S., Wendt, J.B., Zhao, Q., Najork, M.: 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. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.580.
12. Bacciu, D., Errica, F., Micheli, A., Podda, M.: A gentle introduction to deep learning for graphs. In: Neural Networks. 129. pp. 203–221. (2020). https://doi.org/10.1016/j.neunet.2020.06.006.
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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. pp 4171–4186. (2019).
14. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/D14-1179.
15. Karpathy, A., Johnson, J., Fei-Fei, L.: Visualizing and Understanding Recurrent Networks. arXiv:1506.02078 [cs]. (2015).
16. Veličković, P., Casanova, A., Liò, P., Cucurull, G., Romero, A., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings. pp 1–12. (2018).
17. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., Zhang, Z.: Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. arXiv:1909.01315 [cs, stat]. (2020).
18. 1.Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs]. (2017).
19. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. In: The Journal of Machine Learning Research 18, Volume 1. pp 1–52. (2018).