Learning from similarity and information extraction from structured documents

Martin Holeček

Received: 18 October 2020 / Revised: 17 May 2021 / Accepted: 27 May 2021 / Published online: 11 June 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
The automation of document processing has recently gained attention owing to its great potential to reduce manual work. Any improvement in information extraction systems or reduction in their error rates aids companies working with business documents because lowering reliance on cost-heavy and error-prone human work significantly improves the revenue. Neural networks have been applied to this area before, but they have been trained only on relatively small datasets with hundreds of documents so far. To successfully explore deep learning techniques and improve information extraction, we compiled a dataset with more than 25,000 documents. We expand on our previous work in which we proved that convolutions, graph convolutions, and self-attention can work together and exploit all the information within a structured document. Taking the fully trainable method one step further, we now design and examine various approaches to using Siamese networks, concepts of similarity, one-shot learning, and context/memory awareness. The aim is to improve micro $F_1$ of per-word classification in the huge real-world document dataset. The results verify that trainable access to a similar (yet still different) page, together with its already known target information, improves the information extraction. The experiments confirm that all proposed architecture parts (Siamese networks, employing class information, query-answer attention module and skip connections to a similar page) are all required to beat the previous results. The best model yields an 8.25% gain in the $F_1$ score over the previous state-of-the-art results. Qualitative analysis verifies that the new model performs better for all target classes. Additionally, multiple structural observations about the causes of the underperformance of some architectures are revealed, since all the techniques used in this work are not problem-specific and can be generalized for other tasks and contexts.

Keywords One-shot learning · Information extraction · Siamese networks · Similarity · Attention

1 Introduction
The challenge of information extraction is not a new problem. The task has been defined as the transformation of an array of texts into information that can be more readily understood and analyzed. It isolates relevant pieces of text, derives information from them, and then compiles the targeted information into a coherent whole [9].

The explicit category of business documents varies. Existing works on information extraction [25,32,46,49] define them as either “visually rich documents” or “structured” or “semi-structured” documents. In this work, we use the term “structured documents” since the structure of the documents is clear and understandable to a person working in the relevant field, even though the specific structure varies. Moreover, the documents are machine-readable in terms of individual words and pictures (including their positions) on a page, although not “understandable” for a machine with respect to extracting important information.

Classifying all of the information in the financial/accounting industry is important for the “users” of the documents. For example, they may need to know the payment details, amount to be paid, and the issuer information from a collection of documents. In this setting, the input is a document’s page, and the ultimate goal is to identify and output all the words and entities from that page and to classify them by category.

We aim to improve information extraction from business documents and to generally contribute to the field of auto-
mated document processing. Our proposed approach yields a higher success metric compared with previous work and reduces the manual work involved in data entry and/or annotation in the financial/accounting industry.

We focus on the text of business documents including invoices, pro forma invoices, and debit notes, among others. In particular, we target the information that helps in automating various business processes, such as payment on invoices. The typical user of our method would be a medium-sized or larger company that is spending significant time on document processing owing to its size. Although details are scarce in referenced and peer-reviewed works because companies tend to keep their spending information secret, approximations from unofficial (nonscientific) sources (e.g., [34] and [50]) enable estimating company savings. As a hypothetical example, a typical medium-sized company can have approximately 25,000 invoices per month and an improvement of even 1% roughly translates to a savings of more than $500 monthly, which scales with the company size. This potential saving has motivated the topic of information extraction specially in business documents.

1.1 Details and overview

Figure reffig:Example presents one example of an input invoice and output extraction. The documents are clearly not easily understandable inputs. In contrast, an example of trivial inputs would be found in an XML document that has the desired target classes incorporated in a machine-readable way.

With this study, we aim to expand our previous work ([21], referenced as “previous”), in which we showed that neural networks can succeed in the task of extracting important information and even identifying whole, highly specific tables.

The current research question focuses on a “similarity”-based mechanism with various model implementations, and whether they can improve on the existing solution [21]. We hypothesize that a model incorporating similarity techniques will significantly improve the results compared with the existing solution. Moreover, since the presented mechanism is theoretically applicable beyond the scope of document processing, it can be utilized more broadly, whenever it makes sense to include similar known data and apply the query-answer technique.

Ultimately, we present a model along with its source code [22] that outperforms the previous state-of-the-art model. An anonymized version of the dataset is included as an open-source resource, and it represents a notable contribution since its size exceeds that of any other similar dataset to date.

2 Related works

In this subsection, we focus on previous works and approaches in the field of information extraction. The content is heavily based on the text from [21].

A plethora of methods have historically been used for general information extraction. A complete review of these methods, much less comparisons between them, is beyond the scope of the current paper. In general, however, these methods were developed for and evaluated on fundamentally different datasets.

Furthermore, we determined that none of these previous methods is well-suited for working with structured documents (e.g., invoices) because such documents generally do not have any fixed layout, language, caption set, delimiters, fonts, and so forth. For example, invoices vary between countries, companies, and departments, and they change over time. In order to retrieve any information from a structured document, it must first be understood. Our criterion for a reference method is that no human-controlled preprocessing such as template specification or layout fixing is required; we aim for a fully automated and general solution. Therefore, no historical method can serve as a baseline.

Nevertheless, a significant number of recent works do successfully use a graph representation of a document
[8,11,26,32,33,46] and use graph neural networks. Also, a key idea close to the one-shot principle [24,53] in information extraction is used and examined, for example, in [20] and [12]. Both works use notions of finding similar documents and reusing their gold standards (such as already annotated target classes). The latter [12] applies the principle in the form of template matching without the need for any learnable parameters.

2.1 Broader inspiration

More broadly, our approach draws on several research streams pertaining to deep network architectures.

2.1.1 One-shot learning and similarity

A design concept that aims to improve how models contend with new data without retraining of the network is presented in [24,53].

Typically, a classification model is trained to recognize a specific set of classes. In one-shot learning, it is usually possible to correctly identify classes by comparing them with already known data. In contrast to traditional multi-class classification, one-shot learning allows attaining better scores, even with surprisingly low numbers of samples [14]. Sometimes it can even work for classes that are not present in the training set [53].

This concept can help in areas ranging from computer vision variants—from omniglot challenge [28] (also strokes similarity [27]) to object detection [55], finding similar images [57], face detection [51], autonomous vision [19], and speech [13]—and also in the NLP (Natural Language Processing) area [10,29,58].

Among the methods that enable one-shot learning, the most fundamental one utilizes the concept of similarity. Similarity relies on two types of data—“unknown” and “known.” The target values of the known data are already recognized by the method and/or the model. To classify any unknown input, the usual practice is to assign it to the same class as the most similar known input.

Technically speaking, the architecture (typically) contains a “Siamese” part. In particular, both inputs (unknown and known) are passed to the same network architecture with tied weights. We draw inspiration from this basic principle, and we leave more advanced methods of one-shot learning (e.g., GANs [35]) for further research.

For performance reasons, the model is usually not asked to compare new inputs to every other known input—only to a subset. Therefore, a prior pruning technique needs to be incorporated, for example, in the form of a nearest-neighbor search in embedding space, as in [17]. Another option would be to incorporate a memory concept [6] (even in the form of neural Turing machines [48]).

The loss used for similarity learning is called triplet loss because it is applied on a triplet of classes (R reference, P positive, N negative) for each data point:

\[
L(R, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)
\]

where \(\alpha\) is a margin between positive and negative classes, and \(f\) is the model function mapping inputs to embedding space (with Euclidean norm).

Generally speaking, one-shot learning can be classified as a meta-learning technique. For more on meta-learning, we suggest a recent study, like [44] (or just a compiled bibliography online at [36]). Taking the concept one step further yields a concept called “zero-shot learning” [15,37,43].

2.1.2 Other sources of inspiration

Several other sources of inspiration are also meaningfully close to one-shot learning. Since we ask “what labels are similar in the new data,” we need to consider a “query-answer” approach. Recently, the attention principle (i.e., transformer architecture) successfully helped to pave the way to a state-of-the-art performance in language models [45]. It is not uncommon to use attention in one-shot approaches [54] and also in settings related to query answer [16,40,56].

The task of similarity can also be approached as pairwise classification, or even dissimilarity [30].

3 Methodology overview

As we previously argued, every incremental improvement matters. In the current work, we focus on improving the metrics (established by our previous work) by selecting relevant techniques from the field of deep learning. A classical heuristic way to generally improve a target metric is to provide more relevant information to the network. Previous implementations have featured various well-performing techniques that have used the information present in a single invoice, and here we focus on techniques related to similarity.

Since the idea of providing more information is fundamental even for simpler templating techniques [12], we need to stress that, due to the nature of our dataset (available as an anonymized version at [22]), our problem cannot be solved by using templates.

It is important to clarify here the differences between other works and our stream of research (meaning this work and the previous [21]).

The most important difference comes from the dataset that is at our disposal. The dataset explored here is far larger than the datasets used elsewhere and allows for exploring deeper
models as opposed to only using graph neural networks. Indeed, in our previous paper, we proved that graph neural networks work in synergy with additional convolution-over-sequence layers and even global self-attention. Moreover, the dataset quality allowed us to discover (in our previous work) that information extraction and line-item table detection targets do in fact boost each other.

As the current research is focused on deeper models, we will not be using any of the other works as baselines and the commonly used graph neural networks will be incorporated only as one layer amidst many, with no special focus.

We will explore models that could benefit from access to a known similar document’s page. We hope that the model can exploit similarities between documents, even if they do not have similar templates.

In addition, we want to explore the added effect of similarity while keeping everything as close to the previous setting as possible to make sure no other effect intervenes.

The following description mirrors that provided in previous work (in sections 3.3 and 3.4 of [21]). Note that the previous work did not use any means of “similarity” or “nearest pages,” which are introduced in the current work.

### 3.1 Definition of concepts

The main unit of our scope is every individual word on every individual page of each document. Note that other works (e.g., [32]) use the notion of “text segments” instead of “words.” For this work, we define a “word” as a text segment that is separated from the rest of the text by (at least) one white space, and we do not consider any other text segmentation. In general, our approach can also be called “word classification” approach as written in [42], a work where an end-to-end architecture with a concept of memory is explored.

#### 3.1.1 Inputs and outputs

Conceptually, the whole page of a document is considered to be the input to the whole system. Specifically, the inputs are the document’s rendered image, the words on the page, and the bounding boxes of the words. As PDF files are considered, any possible library for reading PDF files can be used for reading the files and getting the inputs. Note that by using any standardized OCR technique, the method could also theoretically be applied to scanned images. (Measuring the effect of OCR errors on the extraction quality is not done here.)

These inputs then undergo feature engineering, as described in 3.2.1, and become inputs for a neural network model.

Each word, together with its positional information, constitutes a “word-box” that is to be classified into zero, one, or more target classes as the output. We are contending with a multi-label problem with 35 possible classes in total. The classes include the “total amount,” tax information, banking information, issuer, and recipient information, among others. (The full set is defined in the code [22].) To obtain a ground truth, the classes were manually annotated by expert human annotators. Interestingly, they had a roughly 3% error rate, which was eliminated by a second annotation round.

### 3.1.2 The dataset and the metric

Overall, we have a dataset with 25,071 documents as PDF files totaling 35,880 pages. The documents are from various vendors and have differing layouts and languages. We split the documents into training, validation, and test sets at random (80%/10%/10%).

The validation set is used for model selection and early stopping. The metric used is computed first by calculating all the $F_1$ scores of all the classes. It is then aggregated by micro-metric principle (more can be found for example in [39]) over all the word-boxes, over all the pages. We then observe and report the scores of the testing set.

The metric choice is inspired by the work [18] in which a content-oriented metric was defined on a character level. In our setting, the smallest unit is a word-box. The choice of the $F_1$ score is based on the observation that the counts of positive samples are outnumbered by the negative samples. In total, the dataset contains 1.2% positive classes.

### 3.2 Shared architecture parts

In the current work, we refer to the architecture from our previous work as a “simple data extraction model.” It serves as one of the baselines here. The architecture of the current model is the same as in the previous work, with the exception of a minor manual parameter tuning. A notable part of the current model, called the “basic building block,” is used in all the new models (defined in Sect. 3.4). Both the simple data extraction model and the basic building block are depicted in Fig. 3.

Since the goal of the overall task and the whole basic building block architecture are shared across all models, by describing the “simple data extraction model,” we also describe all the shared and inherited parts—notably the input and output requirements. We use the full geometrical, visual, and textual information as the input, and the model outputs a multi-class classification for each word-box.

#### 3.2.1 Detailed feature engineering of the inputs

We operate based on the principle of reflecting the structure of the data in the model’s architecture, as machine learning algorithms tend to perform better with this approach.

Springer
The structured information at the input is an ordered sequence of all the word-boxes present on a page. This number can vary by page.

Each word-box has the following features:

– Geometrical:
  – Using a geometrical algorithm, we can construct a directed neighborhood graph over the boxes, which can then be used by a graph CNN (see 3.2.2). Neighbors are generated for each word-box ($W$) by formally assigning every other box to an edge of $W$ that has it in its field of view (being the same $90^\circ$). Then, the closest (center-to-center Euclidean distance) $n$ neighbors are chosen for each side of the box. Our previous results indicated that the optimal number is $n = 1$, and so this number is used in the experiments here. For an example of the constructed directed graph, see Fig. 2. Internally, the graph is saved and passed only as integer indexes denoting the position of each neighbor in a global sequence of word-boxes.
  – We can define a “reading order of word-boxes.” In particular, based on the idea that if two boxes overlap in a projection to the $y$ axis by more than a given threshold (set to 50% in the experiments), they should be regarded as being in the same line from the perspective of a human reader. This not only defines the sequence in which the boxes will be given to the network, but it also assigns a line number and order-in-line number to each box. To get more information, this algorithm can be run again on a $90^\circ$ rotated version of the document. Note that the exact ordering/reading direction (left to right and top to bottom or vice versa) does not matter in the neural network design, thus giving us the freedom to process any language.
  – Each box has four normalized coordinates (left, top, right, bottom) that should be presented to the network.

– Textual:
  – Each word can be presented using any fixed-size representation. Here, we use tailored features common in other NLP tasks (e.g., authorship attribution [7], named entity recognition [38] and sentiment analysis [2]). The features per word-box are the counts of all characters, the counts of the first two and last two characters, length of a word, number of uppercase and lowercase letters, number of text characters, and number of digits. Finally, another feature is engineered to determine whether the word is a number or amount. This feature is produced by scaling and min/maxing the amount by different ranges. (If the word is not a number, this feature is set to zero.) We chose all these features because invoices usually include a large number of entities, IDs, and numbers that the network needs to be able to use.
  – Trainable word features are employed as well, using convolutional architecture over a sequence of one-hot encoded, de-accented, and lowercase characters (“.,+-:%$€£#&’”; all others are discarded). We expect these trainable features to learn the representations of common words that are not named entities.

– Images:
  – Each word-box has its corresponding crop in the original PDF file, where the word is rendered using particular font settings and also has a background. This could be crucial to detect a header or heading, for example, if it contains lines or a different background color or gradient. So for each word-box, the network receives a crop from the original image, offset outwards to be bigger than the text area so the surroundings can also be detected.
Fig. 3  Simple data extraction model. Formally, the whole model consists of two parts: a basic building block and a final classification layer. The basic building block will be used (as a Siamese network) in other models. By removing the final classification layer, we hope to get the best feature representation for each word-box

Each presented feature can be augmented, and we present a random 1% perturbation on coordinates and textual features to regularize the problem and help with generalization.

3.2.2 Simple data extraction model details

To summarize the document’s features described in the previous section, we now explain how they are processed by the model (as Fig. 3 shows). In total, we have five inputs that the neural networks will use:

- Down-sampled picture of the whole document (620 × 877), gray-scaled
- Features of all word-boxes (as defined in the previous section), including their coordinates
- Text as first 40 one-hot encoded characters per each word-box
- Neighbor ids, which are look-up indexes that define the neighboring word-box on each side of the word-box (only one closest neighbor per side is used)
- The integer positions of each word-box defined by the geometrical ordering

In the simple data extraction model, the positions are embedded by positional embeddings (as defined in [31, 52]). An embedding size equal to four dimensions for sin and cos, with a divisor constant of 10,000, is used. The embedded positions are then concatenated with other word-box features.

The image input is reduced by a classical stacked convolution and max-pooling approach. The word-box coordinates (left, top, right, bottom) are not only used as a feature, but also to crop the inner representation of the picture input (see “morphological dilation” in Fig. 3). Finally, we give the model the ability to grasp the image as a whole and supply a connection to the said inner representation, which is flattened and then processed to 32 float features.

Before attention, dense, or graph convolution layers are used, all the features are simply concatenated. To supplement this description, equations and network definitions are given in [22].

As shown in our previous work, all three means of assessing relations between word-boxes are used:

- Graph convolution (also denoted as “GCN”) over the geometrical neighbors of word-boxes is employed to exploit any form of local context. (Details are provided at the end of this section in graph convolution mechanism details.)
- A convolution over sequence layer is a dense layer (or equivalently a 1D convolution layer) applied over the word-boxes ordered by the reading order and allows the network to follow any natural text flow. Implementation-wise, all the word-boxes are ordered in the second dimension at the input (all dimensions being [batch, ordering, feature space]).
- The attention transformer module (from [52]) allows the network to relate word-boxes across the page. Our attention transformer unit does not use causality or query masking.

After these layers are applied, the basic building block definition ends with each word-box embedded in a feature space of a specified dimension, which is 640 unless stated otherwise. The following layer, for the simple data extraction model, is a sigmoidal layer with binary cross-entropy as the loss function. This is a standard setting, since the output of this model is meant to solve a multi-class multi-label problem.

To note implementation detail, batched data fed to the model are padded by zeros per batch (with zero sample weights). Class weights in the multi-task classification problem were chosen (and manually tuned) based on positive class occurrences.

Graph convolution mechanism details The word-box graph (with word-boxes as nodes and neighborhood relation as edges, as depicted in Fig. 2) has a regularity that allows simplifying the graph convolution. First, a small upper bound exists on the number of edges for each node, and second, we do not use any edge classification or specific edge features, in contrast to other works (e.g., [46]). Therefore, we use a simpler implementation than the general form graph convolutions (as in [23, 41]).

More specifically, the implementation uses the generic simplicity present in convolutions at the cost of an additional input. Even a classical convolutional layer over regular picture data can be represented by two basic operations. First, a
gather operation (using tf.gather_nd function from [1]) prepares the data to a regular array (matrix of size number of data points multiplied by the number of data points in one convolutional operation). The second operation is a time-distributed dense layer (equivalently called Conv1D) that simulates the weights of such convolution.

The gather operation needs additional input for each point (pixel or graph node) that specifies the integer indexes of its neighbors (and the node itself). These integer indexes are constructed exactly as stated in 3.2.1.

### 3.2.3 Differences from the previous setting

Just as we have noted the differences from existing research in 2, it is also important to note some detailed differences from our previous work.

**The novelty of this work with regard to the previous setting** Our previous work [21] did not use any nearest-neighbor search or any models that used the notion of similarity or allowed more than one input page at once. In short, our previous work simply laid the fundamental principles of the data, task, and metric and introduced the basic building block (with ablation analysis). Everything that follows this point is new.

**Details changed from the previous setting** Unlike the previous setting, here we do not classify the line-item tabular structures, but only extract (above mentioned) information from the page. In doing so, we demonstrate that the model, despite being optimized on line-item table detection, is versatile. Hence, we make only minor tweaks in the model’s architecture (results of the modifications depicted in Fig. 3).

Previously, two datasets were used for training and validation—“small” (published) and “big” (previously unpublished). The models were tuned on the small dataset, and the big dataset was only used in two experiments to validate that the model scales. In this work, we use the same big dataset. Its previous validation set is split into a new validation set and a new test set to make the test set larger and properly address generalization.

Multiple baselines are employed to prove that the new test set contains documents with layers that are sufficiently different. (The previous work’s test set was small and manually selected.)

### 3.2.4 Differences from one-shot learning

As stated in the introduction, we want to boost the model’s performance for existing target classes by giving the network access to known data (documents) in ways similar to one-shot learning. The main difference is that we are utilizing experiments and architectures that include a fixed selection of classes and/or class information (from the nearest page). Clarifying this detail is important because usually in one-shot learning, no classes are explicitly present in the model—the aim is to generalize to those classes. Our aim, by contrast, is to generalize the model to different and unseen documents with different layouts (instead of classes) that still feature those word-box classes.

### 3.3 The learning framework

The easiest step to boost predictions of an unknown page is to add one more page that is similar and includes word-box classes (annotation) that are already known to the system. That annotation information can then be used in a trained model.

Overall the method works as follows:

- The system needs to keep a notion of already known documents in a reasonably sized set. We call them “known” because their classes/annotations should be ready to use.
- When a “new” or “unknown” page is presented to the system, it searches for the page that is most similar to the “known” pages (given any reasonable algorithm).
- The model is allowed to use all the information from both pages (and “learn from similarity”) to make the prediction.

The system can then even present the predictions to a human for verification and then add the page to the existing database of known pages. However, we do not explore the database size effects here.

Before making predictions, the incorporated model should be trained on pairs of pages to simulate this behavior.

In this process, there are multiple points to be examined, but we posit that the most interesting research question is the following:

Holding all other factors fixed (meaning the train/test/validation split, evaluation metrics, data format, and method for searching for a similar page), what approach and what neural network architecture are able to raise the test score the most?

We argue that this is the right question to ask since all other factors usually have a known effect on the result if best practices are followed. As an example, we note that bigger datasets typically yield better scores; the presence of more “nearest neighbors” typically has a boosting effect similar to ensembling, and so on.

Further, from a practical point of view, only two pages can fit into a single GPU memory with all the features described before.

As already stated earlier, we draw inspiration from the one-shot learning framework. For predicting an unknown page, we define a way to search for one “nearest” known page and allow the model access to its annotations as known target classes. Note that not all explored models use the nearest known page. In addition to the simple data extraction model,
we consider some baselines that do not require the nearest page to verify the assumptions.

### 3.3.1 Nearest-neighbor definition

For one-shot learning to work on a new and unknown page (sometimes denoted as the “reference”), the system always needs to have a known (also denoted as “similar” or “nearest”) document with known annotations at its disposal. Since focusing on that task properly is beyond the scope of the current paper, we have used the nearest-neighbor search in the space of the page’s embeddings to select only one closest page of a different invoice document.

The embeddings were created through a process similar to a standard one, as described in [5]. We used a different model (older and proprietary) that was trained to extract information from a page. To change the classification model into an embedding model, we removed its latest layer and added a simple pooling layer. This modified the model to output 4850 float features based only on image input. These features were then assigned to each page as its embedding.

We then manually verified that the system would group similar, or at least partially similar, pages near each other in the embedded space.

These embeddings were held fixed during training and inference and computed only once in advance.

### 3.3.2 Constraints of nearest-neighbor search

We want the trained model to behave as close to the real world as possible, so the nearest page search process needs to be constrained. Each document’s page can select the nearest annotated page only from the previous documents in a given order. As in a real service, we can only see the received and processed documents.

In addition, we want the method to be robust, so before each epoch, the order of all pages is shuffled and only the previous pages (in the given order) from a different document are allowed to be selected.

This holds for all sets (training, validation, and test) separately. To verify the consistency of this strategy, some experiments are tweaked by the following variations:

- Allowed to additionally use the training set as a data source for the “nearest annotated” input. We expect the performance to rise.
- Made “blind” by selecting a random document’s page as the nearest known input. We expect the performance to fall.

### 3.3.3 Baselines

To challenge our approach from all possible viewpoints, we consider multiple baselines:

1. To use only the simple data extraction model (Sect. 3.2 and Fig. 3) without any access to the nearest known page.
2. “Copypaste” baseline. This model will only take the target classes from the nearest page’s word-boxes and overlay them on the new page’s word-boxes (where possible). We expect a low score since the documents in the dataset are different, and this operation will not copy anything from any nearest page’s word-box that does not intersect with a new page’s word-box. This approach uses no trainable weights and is the simplest example of a templated approach that does not have hard-coded classes.
3. “Oracle” baseline. This model will always correctly predict all classes that are present in the nearest page. We use this model to measure the quality of the nearest-page embeddings to gain additional insight into the dataset’s properties. The metric used for this model is not $F_1$, but a percentage of all word-boxes that can be classified correctly. The score is expected to be only moderately good, as the embeddings are created in a rather unsupervised manner (regarding their usage). We want to explore a different influence than that already explored by existing works aimed at finding the best helping pages [12]. Ultimately, we want to present a model that can work even if the quality of the embeddings is just moderate.
4. Fully linear model with access to concatenated features from both new and known pages. This model does not feature picture data.

The choice of baselines (and ablations later in experiments) helps to verify and demonstrate multiple claims:

- The newly proposed models can beat the previous results (which is achieved if the simple data extraction model is beaten).
- The documents are different enough.
- A similarity search alone is not enough, even if the embeddings have better-than-moderate quality with regard to the similarity.
- To justify the complexity of models presented in the following section, 3.4.

All baselines and all models presented in the current work will have the same desired output—they will provide the multi-class classification for each word-box.
3.4 Model architectures

We have described the basic information extraction block that aims to output the best trained latent features for each word-box. All the model architectures incorporate this block used as a Siamese network for the inputs of both unknown and known pages. Each architecture is trained as a whole, no pre-training or transfer learning takes place, and every model is always implemented as a single computation graph in Tensorflow.

We explore multiple different architectural designs for predicting the targets (at their outputs) by using the closest nearest page from already annotated documents.

1. “Triplet Loss architecture”—using Siamese networks “canonically” with triplet loss.
2. “Pairwise classification”—using a trainable classifier pairwise over all combinations of word-box features from reference and nearest page.
3. “Query-answer architecture” (or “QA” for short)—using the attention transformer as an answering machine to a question of “which word-box class is the most similar.”

The copypaste baseline represents a reasonable basic counterpart for triplet loss and pairwise classification. The fully linear model represents the simplest counterpart for the query-answer approach, which also has all the classes hard-coded.

There is a slight distinction between the first two architectures and the third. In QA architecture the class is a direct prediction of the network for each word-box. In triplet loss and pairwise classification, the models predict (for each unknown word-box) all the similarities to all the word-boxes from the known page. All the similarity values then collectively determine the target class for the word-box.

Since the embeddings used to search for the nearest page are not ideal, the models may not be able to predict some classes. To assess these methods fairly, we scale the metrics used to measure the success by the performance of the corresponding oracle baseline (defined in refsubsec:Baselines). Or put differently, we do not count errors that the model cannot predict correctly owing to some classes being absent from the nearest page. This reflects our aim to explore the effects of the models that can operate with the nearest page.

In reality, if these (triplet loss and pairwise classification) methods prove to be the most efficient, the hyperparameters, such as the quality of the embeddings (or the number of the nearest pages), would need to be addressed to overcome the score of the previous results. A perfect performance of the scaled metric means that the extraction is only as good as the oracle baseline.

In the experimental result Sect. 4, we include a test of triplet loss and pairwise classification models that makes them predict a different and possibly easier target. Instead of “do these two word-boxes have the same target class,” the easier testing target is “do these two word-boxes have the same length of text inside.” This test is meant to show that the method is well-grounded and usable for any other reasonable target definition.

In comparison, QA architecture has the classes hard-coded in the design, which means it can predict a class not present in the nearest page. Therefore, no metric scaling is necessary in the evaluation of the QA model.

3.4.1 Triplet loss architecture

Since our data point is a word-box, strictly adhering to the use of triplets of word-boxes for triplet loss would mean executing the model for each word-box pair once. To avoid impairing the performance (as there can be as many as 300 word-boxes per page) and/or losing in-page dependencies, the proposed architecture (see Fig. 4) features a mechanism of tiling and filtering to pass all combinations of word-boxes at once.

The filtering mechanism filters out all but the annotated word-boxes from the nearest page. It eliminates most of the unused information and, in doing so, saves memory and computation time. The tiling mechanism takes two sequences—first, the sequence of reference page word-boxes, and second, the sequence of nearest page filtered word-boxes. It subsequently produces a bipartite matrix. The model is then able to compute pairwise distances between the same and different classes. These distances are then used for triplet loss computation (see mathematical definition in the section below).

Additionally, we can include a single classification layer to be automatically calibrated on the distances, which adds
a binary cross-entropy term to the loss. The loss is averaged over all the word-boxes to account for the fact that no word-box is ever alone on a page.

We rely on the (manually verified) fact that during training each page has more than one class annotated. Consequently, there are always positive and negative samples present, as there should be in the triplet loss.

There are three possible modifications to explore:

- Adding annotated class information to the nearest page’s features.
- Using a “loss-less triplet loss,” which is a loss similar to the triplet loss but without the min–max functions (see definition below).
- Modifying the distance and/or loss computations by means of constants or by using cosine similarity instead of Euclidean space.

### 3.4.2 Triplet-loss inspired losses

The purpose of this model is to use the triplet loss in the most straightforward manner in our setting. The only mathematically interesting description to be given here is the triplet loss and “loss-less triplet loss” defined over word-boxes since all trainable layers in this model (and binary cross-entropy loss) are defined in referenced works.

In traditional triplet loss, positive, negative, and reference samples are necessary. Since we need to account for a whole page full of word-boxes, we must compute all combinations at once.

We denote the quantity \( \text{truth\_similar}(i, j) \) to indicate if the word-boxes \( i, j \) (\( i \)-th being from the unknown page, \( j \)-th being in the nearest page) share the same ground truth class \((1.0 = \text{yes}, 0.0 \text{ otherwise})\). Next we define \( \text{pred\_dist}(i, j) \) as the predicted distances between the feature spaces of the word-boxes by the model. Then, we can calculate two loss variants (“triplet_like” and “loss-less”) inspired by triplet loss as follows:

\[
\begin{align*}
\text{pos\_dist}_{i,j} &= \text{truth\_similar}(i, j) \cdot \text{pred\_dist}(i, j) \\
\text{neg\_dist}_{i,j} &= (1.0 - \text{truth\_similar}(i, j)) \cdot \text{pred\_dist}(i, j) \\
\text{triplet\_like} &= \max_{i,j}(0, \alpha + \max(\text{pos\_dist}_{i,j}, \text{neg\_dist}_{i,j})) \\
\text{lossless} &= \sum_{i,j} \text{pos\_dist}_{i,j} - \sum_{i,j} \text{neg\_dist}_{i,j}
\end{align*}
\]

(The equations are present in a form to be most similar to the source code, i.e., not simplified.) The quantities \( \text{pos\_dist} \) and \( \text{neg\_dist} \) are just helper variables to reveal the similarity with the original triplet loss, and \( \alpha \) is a parameter of the same meaning as in the original triplet loss. The two new losses represent two different approaches used in the reduction from a matrix to a single number. We can either take the largest positive and negative values and use them in the triplet loss equation, or we can sum all the positive and negative terms. The real difference is how the gradients are propagated; variants with min/max always propagate fewer gradients than the former per gradient-update step in the training phase. All the losses can be used at once with a simple summation.

The name “loss-less” comes from the idea described in [3]. To our knowledge, it does not occur in any other scientific work beyond this online article.

Finally, we present different options for the loss terms. Since we focus on different architectures and not on hyperparameters, we omit from this description the specific constants used to sum the terms. In the experiment Sect. 4, we present the best results that we were able to achieve by manual hyperparameter tuning. The results of the tuning and various options are clearly defined in the accompanying code [22] together with all the specific hyperparameters.

### 3.4.3 Pairwise classification

Pairwise classification architecture (see Fig. 5) uses the same tiling and filtering mechanism as described in 3.4.1. But instead of being projected into a specific feature space to compute distances, the data points are simply “classified” by using a traditional approach of sigmoidal activation function and binary cross-entropy loss.

As in our previous model, we have the option of adding annotated class information to the nearest page’s features. We have also explored various sample weight options and an optional “global refinement” section. The optional refinement pools information from each word-box uses a global transformer and propagates the information back to each reference word-box.
3.4.4 Query-answer architecture

In the heart of the QA architecture (see Fig. 6) lies the fact that the transformer module with three inputs can be used as a query-answer machine.

More variants could be explored here:

- “Query all”: Does it help if the transformer can query not only the nearest page’s word-boxes, but also those of the new page itself?
- “Skip connection”: Would a skip connection to the base information extraction block improve the performance?
- “Filter”: Should it filter only annotated word-boxes from the nearest page (as in the two previous approaches)?
- “Field of view”: Would adding a field of view information flow from the new page’s word-boxes to the nearest page make a difference?

Technically a field of view is realized by providing indexes, in which word-boxes would be close to each other by geometrically projecting each word-box from the reference page to the annotated page and selecting a fixed number of Euclidean-closest word-boxes. The limits for the distances were chosen based on average distances between word-boxes of the same class on different documents. The loss used for this model is classical binary cross-entropy.

The main idea of this architecture is a query-answer mechanism and so it can be applied in any different setting with Siamese networks.

4 Experiments and results

In this section, we present the results for each group of experiments. An Adam optimizer was used together with an early stopping parameter of 20 epochs (to maximally 200 epochs). The average time was 40 min per epoch on a single GPU. The baseline needed only 10 min per epoch (since it did not need any “nearest” page mechanism). The model selected in each experimental run was always the one that performed the best on the validation set in terms of loss.

The basic building blocks present in every architecture were usually set to produce feature space of dimensionality 640 (unless noted otherwise in the tables as “feature space $n$”).

4.1 Baseline results

We report some variations of architecture parameters for the simple data extraction model (introduced in Sect. 3.2) in Table 1. The goal is to show how sensitive the basic model is to various changes and to tune the baseline for extracting the classes.

The results could be interpreted as the model reaching its maximal reasonable complexity at one transformer layer and smaller feature space. As we will see, this does not apply to the Siamese settings as the gradients propagate differently when parts of the architecture have tied weights.

| Previous state of the art, re-tuned (and possible notable tweaks, see Sect. 4) | Test micro F1 score |
|---|---|
| 2x attention layer, feature space 640 | 0.6220 |
| 1x attention layer, feature space 640 | 0.8081 |
| 1x attention layer, feature space 64 | **0.8465** |
| 1x attention layer, f. space 64, fully anonymized | 0.6128 |
| 1x attention layer, f. space 64, only text features | 0.7505 |

The bold number indicates the best achievable results of our previous work alone, therefore it is the score we aim to beat in this article.
To beat our previous state-of-the-art results, we need to improve the $F_1$ score to exceed 0.8465, which is the best score for the simple data extraction model.

### 4.1.1 Copypaste baselines

Table 2 shows the fairly low score of those simple baselines. Such a low score illustrates the complexity of the task and variability in the dataset. Simply put, it is not enough to just overlay a different similar known page on the unknown page because the dataset does not contain completely identical layouts.

We can also see that an important consistency principle holds for the nearest neighbors:

- Selecting a random page decreases the score.
- Using a bigger search space for the nearest page increases the score.

### 4.1.2 Oracle baseline

Table 3 displays the “moderate quality” of the embeddings. Specifically, only roughly 60% of word-boxes have their counterpart (class-wise) found in the nearest page.

When the nearest-neighbor search is replaced with a completely random pick, an interesting property of the dataset emerges in that the number of word-boxes that have a similar class on the random page increases a little. This is because the distribution of class presence in the pages is skewed, which is explained by vendors usually wanting to incorporate more information into their business documents.

### 4.1.3 Linear baseline

The linear model has attained 0.3085 test micro $F_1$ score. Its performance justifies the progress from the basic copypaste model toward trainable architectures with similarity.

### 4.2 Results of architectures with similarity

In this section, we consider all the designed architectures that compete with the baselines.

The results for triplet loss architecture are presented in Table 4, and the results for pairwise classification are in Table 5.

Both pure triplet loss approaches and pairwise classification performed better than simple copypaste, but still worse than linear architecture. We suggest two possible reasons for this outcome:

- The existence and great prevalence of unclassified (uninteresting) data in the documents.
Table 6 Experimental results of QA architecture

| Experiments architecture (and possible notable tweaks, see Sect. 3.4.4) | Test micro $F_1$ score |
|-------------------------------------------------|----------------------|
| All QA improvements in place                   | 0.9290               |
| Fully anonymized dataset                       | 0.7078               |
| Only text features                              | 0.8726               |
| Nearest page set to random                      | 0.8555               |
| Without field of view                          | 0.8957               |
| Without query all                              | 0.7997               |
| Without skip connection                        | 0.9002               |
| Without filtering                               | 0.8788               |

This reason is supported by the fact that all methods with hard-coded class information (including simple linear baseline) scored better. Unfortunately, this phenomenon could be specific to the dataset. We could not replicate the suboptimal results by modeling this situation in an existing and otherwise successful task (omniglot challenge) by adding non-classifiable types and by increasing the percentage of negative pairs.

- Missing connections to the unknown page.

Table 6 shows how the score drops in QA architecture when we switch to the variant “without query all.” We conclude that even the best architecture needs a meaningful information flow from the reference page itself and not only from the nearest page. That information flow is missing in triplet loss and pairwise classification.

To gain more insight, we tested the architectures on a different target value, which was defined as “does the text in these word-boxes have the same length.” In this setting, the architectures achieved a significantly higher score of 0.7886. This supports our theory that the unclassified data (see above) was responsible for the underperformance of triplet loss and pairwise classification, since all data in the document were useful for the text lengths target.

4.2.1 Query answer

The query-answer architecture scored the best, with a micro $F_1$ score of 0.9290 with all the proposed architectural variants employed at once. In Table 6, we present an ablation study, showing that each of the components (field of view, query all, skip connection, filter, nearest search as defined in 3.4.4) related to QA architecture is clearly needed, as the score drops if any single one is turned off.

Compared with the previous model Table 1, an improvement of 0.0825 in the $F_1$ score is achieved. Also, the experiment on the anonymized dataset and the dataset with only text features shows that the architecture is versatile enough to not fail the task and to show similar improvement in the score on the anonymized dataset (by 0.0950). It also verifies that all the visual, geometric, and textual features are important for good quality results.

4.2.2 Qualitative comparison

We conclude with more qualitative analysis, specifically, a comparison of the best QA model and the simple data extraction model.

To start, we select pages from a random subset of the test set and present example visualizations in Figs. 7, 8 and 9 to illustrate a manual inspection of prediction visualizations. They show the best prediction from the query-answer model (Fig. 7), the worst prediction from the query-answer model (Fig. 8), and finally the worst prediction of the simple data extraction model (Fig. 9). A successfully classified word-box is a true positive, while successfully classified unimportant text is a true negative. Misclassifications of true positives (“miss”) and true negatives (“extra”) are also indicated.

Both the simple data extraction model and the QA model have examples of pages that look like results in Fig. 7 and are 100% perfectly extracted (or classified). However, the results vary in the worst cases, which is why examples from both models are presented in Figs. 8 and 9.

Motivated by this difference, we can look at which classes both models extract best and worst. Those scores are presented in Table 7.
This detailed inspection shows that both models excel at classes that usually appear together (but not in any fixed layout or order) in business documents. Those classes are all the recipient information (DIC, IC, Spec symbol) and the sender information. Moreover, recipient information is usually required information on an invoice, and thus, it is the most frequent class and the network easily excels at detecting it.

Interestingly, page numbering could be seen as an easy class to classify, but the previous model actually classified it with a very low score. The score jumps to a very high value when we switch to the QA model though. One possible reason for this improvement is that the page number usually appears alone somewhere near an edge of the page. Thus, its nearest word-boxes are random and might cause confusion for the GCN module and for convolution over the sequence as well. When a similar page is presented to the model, the score jumps higher possibly because the nearest page might have page numbering in a similar position.

The QA model, as a possible improvement from our previous results, holds an important property we desire. In particular, we have verified that the score for all classes has increased uniformly by at least 0.02 points (median gain being 0.04), even for the previously best-performing classes. This property is important to verify, since the QA architecture incorporates the simple data extraction model, and we expect it to “fall back” to it when the nearest page does not provide enough information. If this fallback does not happen, some gradients would not be propagated correctly.

The improvement of some fields by only roughly 2% may be seen as a small improvement. But in reality (as stated in the introduction), the 2% improvement translates into less time and effort for companies processing more than 500 invoices per month. This reduced time and effort translates to more than $1000 of savings per month as well as a reduction in the company’s carbon footprint.

### 5 Conclusions

Multiple baselines were provided and evaluated to gain more knowledge about the data and establish the need for bigger and more complicated models.

We have designed multiple ways to incorporate similarity and memory—in terms of access to existing data—into the existing data extraction model, and we studied the gains of
various models in a fixed setting. The successful gain was an 8.25% increase in the $F_1$ score compared with our previous results by using “query-answer” inspired architecture. By the referenced heuristics, this improvement translates roughly into $4000$ dollars savings of manual work per month for a medium-sized company.

We have verified that all possible parts of the architecture are needed in the training and prediction of the QA model to achieve the highest score. Moreover, the improvement holds even in the case of the anonymized dataset.

In a qualitative analysis of the results, we showed that the score improvement is meaningful across all the classes. Furthermore, the solution was shown to significantly boost the previously most-problematic classes.

For the other models that underperform (as triplet loss and pairwise classification), we have identified the possible cause as most words on the page not belonging to any class, and we supported the hypothesis with an additional experiment.

Further work could incorporate a way to create some artificial classes and measure the supposed increase in score for the triplet and pairwise classification models.

From a quantitative point of view, an opportunity exists to explore and improve extraction scores by tuning all the possible parameters of the system, namely the number of the nearest pages used and the quality of the page embeddings. The page embeddings can possibly be jointly trained with the word-box classifier.

Qualitatively, we identify some possible new research questions:

- What is the effect of the size of the datasets? By exploring the effect of the size of the training dataset and/or the search space for the nearest pages, we could ask if (and when) the model needs to be retrained and what a sample of a difficult-to-extract document looks like.
- How to improve the means of generalization? Currently, the method generalizes to unseen documents. In theory, we could desire a method to generalize to new classes of words because in the current approach, the model needs to be retrained if a new class is desired to be detected and extracted.

In practice, our solution has a particular strength that transforms these two points from potential hurdles to interesting research questions. The model can fit into just one consumer-grade GPU and trains from scratch within 4 days at most using only one CPU process. Compared with recent state-of-the-art NLP methods that take ample resources to train (such as [4]), our model can be retrained and/or fine-tuned for any particular use-case quickly and effectively (even more with transfer learning techniques). Thus, any assortment of problems are solved by the industrial standards [5].

As a part of this work, the dataset and source codes are published in [22]. This resource should enable wider research of deep learning models for information extraction because it was previously impossible for researchers to collect a dataset of this size and quality.

Acknowledgements The Rossum.ai team deserves thanks for providing the data and background that enabled the development and growth of this work.

Author Contributions The principal author is responsible for the study concept and design, execution, coding, and research. The rest of the Rossum team is responsible for data acquisition, annotation, and storage and for the creation of a working product and environment that enabled a scientific study of this scope.

Funding This work was supported by the Grant SVV-2020-260583. Partial financial support was received from Rossum and Charles University.

Data availability An anonymized version of the dataset is publicly available at [22], together with all the codes. The improvement on previous results can be reproduced using the anonymized data without disclosing any sensitive information.

Code availability The source codes are publicly available from a GitHub repository [22].

Declarations

Conflict of interest The author (Martin Holeček) has received financial support from Rossum and from Charles University, where he is currently pursuing a PhD. The author has an employment and/or contractual relationship with Rossum, Mediacle, and AMP Solar Group.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org (2015). https://www.tensorflow.org/

2. Abbasi, A., Chen, H., Salem, A.: Sentiment analysis in multiple languages: feature selection for opinion classification in web forums. ACM Trans. Inf. Syst 26(3), 1–34 (2008). https://doi.org/10.1145/1361684.1361685

3. Arensault, M.O.: Lossless triplet loss (2018). https://towardsdatascience.com/lossless-triplet-loss-7e932f990bf2

4. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)

5. Burkov, A.: Machine Learning Engineering. True Positive Incorporated (2020)

6. Cai, Q., Pan, Y., Yao, T., Yan, C., Mei, T.: Memory matching networks for one-shot image recognition. In: Proceedings of the IEEE
Conference on Computer Vision and Pattern Recognition, pp. 4080–4088 (2018)
7. Chen, Z., Huang, L., Yang, W., Meng, P., Miao, H.: More than word frequencies: authorship attribution via natural frequency zoned word distribution analysis. CoRR (2012), arXiv:1208.3001
8. Coudronnion, B., Lemaître, A.: Recognition of Tables and Forms, pp. 647–677. Springer, London (2014). https://doi.org/10.1007/978-0-85729-859-1_20
9. Cowie, J., Lehner, W.: Information extraction. Commun. ACM 39, 80–91 (1996)
10. Dalvi, B.B., Cohen, W.W., Callan, J.: Websets: extracting sets of entities from the web using unsupervised information extraction. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, pp. 243–252 (2012)
11. d’Andecy, V., Hartmann, E., Rusinol, M.: Field extraction by hybrid incremental and a-priori structural templates. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 251–256 (2018). https://doi.org/10.1109/DAS.2018.29
12. Dhakal, P., Munikar, M., Dahal, B.: One-shot template matching for automatic document data capture. In: 2019 Artificial Intelligence for Transforming Business and Society (AITB), vol. 1, pp. 1–6 (2019). https://doi.org/10.1109/AITB48515.2019.8947440
13. Eloff, R., Engelbrecht, H.A., Kamper, H.: Multimodal one-shot learning of speech and images. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8623–8627. IEEE (2019)
14. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. 28(4), 594–611 (2006)
15. Felix, R., Sasselli, M., Reid, I., Carneiro, G.: Multi-modal ensemble classification for generalized zero shot learning. arXiv preprint arXiv:1901.04623 (2019)
16. Galassi, A., Lippi, M., Torroni, P.: Attention in natural language processing. Computation and Language (2020)
17. Ghosh, S.K., Valveny, E.: R-phoc: segmentation-free word spotting using CNN. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 801–806. IEEE (2017)
18. Göbel, M., Hassan, T., Oro, E., Orsi, G.: ICDAR 2013 table competition. In: In 2013 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 1449–1453. IEEE (2013)
19. Grigorescu, S.M.: Generative one-shot learning (GOL): a semiparametric approach to one-shot learning in autonomous vision. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7127–7134. IEEE (2018)
20. Hamza, H., Belaid, Y., Belaid, A.: Case-based reasoning for invoice analysis and recognition. In: Weber, R.O., Richter, M.M. (eds.) Case-Based Reasoning Research and Development, pp. 404–418. Springer, Berlin (2007)
21. Holecek, M., Hoskovec, A., Baudis, P., Klinger, P.: Table understanding in structured documents. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), vol. 5, pp. 158–164 (2019). https://doi.org/10.1109/ICDARW.2019.40098
22. Implementation details for this work, source codes and curated anonymized dataset to reproduce results. https://github.com/Darthholi/similarity-models
23. Jean-Pierre Tixier, A., Nikolentzos, G., Meladianos, P., Vazirgiannis, M.: Graph Classification with 2D Convolutional Neural Networks. arXiv e-prints arXiv:1708.02218 (2017)
24. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol. 2. Lille (2015)
25. Kosala, R., Van den Bussche, J., Bruynooghe, M., Blockeel, H.: Information extraction in structured documents using tree automata induction. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Principles of Data Mining and Knowledge Discovery, pp. 299–311. Springer, Berlin (2002)
26. Krieger, F., Drews, P., Funk, B., Wobbe, T.: Information extraction from invoices: a graph neural network approach for datasets with high layout variety (2021)
27. Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 33 (2011)
28. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: The omniglot challenge: a 3-year progress report. Curr. Opin. Behav. Sci. 29, 97–104 (2019)
29. Lampinen, A.K., McClelland, J.L.: One-shot and few-shot learning of word embeddings. CoRR (2017). arXiv:1710.10280
30. Lin, Z., Davis, L.S.: Learning pairwise dissimilarity profiles for appearance recognition in visual surveillance. In: Advances in Visual Computing, pp. 23–34. Springer, Berlin (2008)
31. Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., Yosinski, J.: An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. arXiv e-prints arXiv:1807.03247 (2018)
32. Liu, X., Gao, F., Zhang, Q., Zhao, H.: Graph convolution for multimodal information extraction from visually rich documents. arXiv preprint arXiv:1903.11279 (2019)
33. Lohani, D., Abdel, B., Belaid, Y.: An Invoice Reading System using a Graph Convolutional Network. In: International Workshop on Robust Reading. PERTH, Australia (2018). https://hal.inria.fr/hal-01960846
34. Manual typing is expensive: The tco of invoice data capture (part 2). https://rossum.ai/blog/manual-typing-is-expensive-the-tco-of-invoice-data-capture-part-2/
35. Mehrotra, A., Dukkipati, A.: Generative adversarial residual pairwise networks for one shot learning. arXiv preprint arXiv:1703.08033 (2017)
36. Meta learning papers. https://github.com/floodsung/Meta-Learning-Papers
37. Mishra, A., Krishna Reddy, S., Mittal, A., Murthy, H.A.: A generative model for zero shot learning using conditional variational autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 2188–2196 (2018)
38. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes 30(1), 3–26 (2007)
39. Narasimhan, H., Pan, W., Kar, P., Protopapas, P., Hamaswamy, H.G.: Optimizing the multiclass f-measure via biconcave programming. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 1101–1106. IEEE (2016)
40. Nie, Y.P., Han, Y., Huang, J.M., Jiao, B., Li, A.P.: Attention-based encoder–decoder model for answer selection in question answering. Front. Inf. Technol. Electron. Eng. 18(4), 535–544 (2017)
41. Niepert, M., Ahmed, M., Kutzkov, K.: Learning Convolutional Neural Networks for Graphs. arXiv e-prints arXiv:1605.05273 (2016)
42. Palm, R., Laws, F., Winther, O.: Attend, copy, parse end-to-end information extraction from documents. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 329–336 (2019)
43. Paul, A., Krishnan, N.C., Munjal, P.: Semantically aligned bias reduction zero shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7056–7065 (2019)
44. Peng, H.: A comprehensive overview and survey of recent advances in meta-learning. arXiv:2004.11149 (2020)
45. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog 1(8), 9 (2019)
46. Riba, P., Dutta, A., Goldmann, L., Fornes, A., Ramos, O., Llados, J.: Table detection in invoice documents by graph neural networks, pp. 122–127 (2019). https://doi.org/10.1109/ICDAR.2019.00028
47. Rossum’s blogpost “extracting invoices using ai” at medium.com. https://medium.com/@bzamecnik/extracting-invoices-using-ai-in-a-few-lines-of-code-96e412df7a7a
48. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: One-shot learning with memory-augmented neural networks. arXiv preprint arXiv:1605.06065 (2016)
49. Smith, D., Lopez, M.: Information extraction for semi-structured documents. In: Proceedings of the Workshop on Management of Semistructured Data (1997)
50. Tenhunen, M., Penttinen, E.: Assessing the carbon footprint of paper vs. electronic invoicing (2010). https://aisel.aisnet.org/acis2010/95
51. Thakurdesai, N., Raut, N., Tripathi, A.: Face recognition using one-shot learning. Int. J. Comput. Appl. 182, 35–39 (2018). https://doi.org/10.5120/ijca2018918032
52. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All You Need. arXiv e-prints arXiv:1706.03762 (2017)
53. Vinyals, O., Blundell, C., Lillicrap, T.P., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. CoRR (2016). arXiv:1606.04080
54. Wang, P., Liu, L., Shen, C., Huang, Z., van den Hengel, A., Tao Shen, H.: Multi-attention network for one shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2721–2729 (2017)
55. Xu, L., Wang, Y., Li, X., Pan, M.: Recognition of handwritten Chinese characters based on concept learning. IEEE Access 7, 102039–102053 (2019)
56. Yang, Z., He, X., Gao, J., Deng, L., Smola, A.: Stacked attention networks for image question answering. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 21–29 (2016)
57. Yim, J., Kim, J., Shin, D.: One-shot item search with multimodal data. arXiv:1811.10969 (2018)
58. Yin, W.: Meta-learning for few-shot natural language processing: a survey. arXiv:2007.09604 (2020)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.