Extracting Structured Information from Invoices in ForceZBO

Casper van Aarle

November 25th, 2020

Contents

1 Introduction

The digitization of many communication forms causes the paper letter to be replaced by a computer screen. Although digitizing started in the 1950s, and it has had a clear acceleration with the introduction of the world wide web, still not all transmissions are digital. Invoices and receipts are printed on paper and distributed as a form of communication. For example, healthcare insurance companies are relying on those documents to correctly reimburse purchases.

Nowadays, insurance companies are trying to digitize the total stream of invoice communication (for example the Dutch Vecozo-standard), but still a substantial part of invoice communication systems consist of sending a copy of the printed document. Before the insurance company can either accept or decline the request, it must extract the correct information from those documents, which is called the 'invoice extraction' task.

ForceZBO is a system, which is designed by Topicus in correspondence with and commissioned by an insurance company, that implements software handling invoice extraction manually, see figure 1. Big insurance companies may deal with thousands of daily incoming scanned invoices which they have to process manually. Due to the sheer volume of invoices that must be processed, there is a clear need for automating this task. In the current environment, ForceZBO, no automatic invoice processing is taking place.

This assignment, which was given by Topicus, includes (1) doing research on the feasibility of information extraction in the ForceZBO system and (2) implementing such a system. In the next paragraphs, we describe the contributions made by this paper regarding the automatic invoice extraction process of ForceZBO.

There is some research available on the automatic extraction of invoices, most of which incorporate machine learning as a big part of the implementation. We will be focusing on describing possible invoice extraction methods.

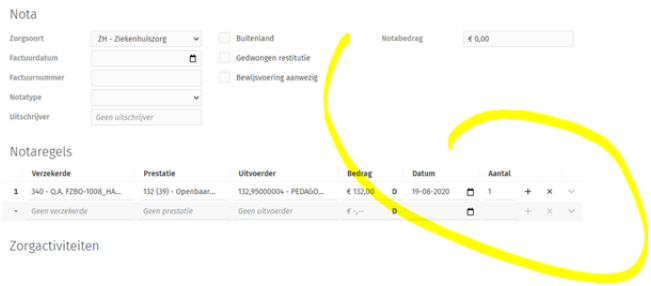

Figure 1: Module to fill in the information from a corresponding invoice. Source: Topicus

Invoices are privacy-sensitive documents, as there will always be some personal information incorporated in the invoice. Normal machine learning implementations would require a big set of invoices, and for various reasons including privacy restrictions, such a set may not always be available in this stage of the research. But traditional machine learning methods are trained with big amounts of data available, and cannot handle information that comes in sequentially over time. It can only remember information from

one learning session. We will be discussing possible solutions to this problem.

Experiment 1 will focus on testing two versions of a self-built model inspired by the literature on previously seen templates. A seen template is an invoice layout that has already been learnt by the model. An unseen template is a layout unknown to the model.

Testing is an important part of building implementations, as you need to see if your results improve when a change has been made. Performance can be measured across seen and unseen templates. In experiment 1 we test over templates that were already learnt by the model to see if it remembered the layouts. However, how it performs on unseen templates is quite a different problem, as the model has to generalize much more of its knowledge. In experiment 2, we compare our self-built implementation to an existing implementation to see the performance on unseen invoices. We have not made any implementation in ForceZBO as the requirements to deploy such a system were not met. We elaborate on this in section 2.3.

As a summary, we come to the following list of contributions of this report:

1. Literature research on the current concepts used for invoice extraction.
2. Short literature research on the possibilities to learn sequentially.
3. Comparison of two versions of the self-built implementation to apply invoice extraction on invoices.
4. Comparison of self-built and existing implementation on unseen templates.

The paper will provide an overview of different concepts in the Background section necessary to understand recent work on invoice extraction. The Concepts section provides the main tools used in those approaches and gives an in-depth view of the workings of different approaches. It also gives a brief introduction to sequential learning methods. The Experiments section presents our research on building a custom implementation for invoice extraction, and the application of the model on unseen templates. We discuss results, draw conclusions from these, and give a recommendation for Topicus to continue its research on invoice extraction.

2 Background

Before diving into recent research papers on invoice extraction, it is important to know how the ForceZBO system handles invoices, what a machine learning approach looks like, and what knowledge is needed before considering possible solutions.

2.1 ForceZBO Fields

To process a reclamation request, every invoice requires the correct information in the provided fields. 'Fields' are the input slots that are filled by the user. Different kinds of fields may be filled with different data types. We sometimes refer to the 'ForceZBO library' that includes all the possible labels for some fields. We consider three kinds of fields:

1. **Category Fields:** Some fields have a limited set of input options. Those include the fields: 'Zorgsoort', 'Notatype', 'Buitenland', 'Gedwongen restitutie', 'Bewijsvoering aanwezig' which can be seen in figure 1. These fields are characterized by input that only consist of a few options. In most cases, those fields do not occur explicitly in the invoice.

2. **Autocomplete Fields:** Then there is the second type of input field, with ForceZBO library objects as possible input. The user must provide the full or partial input (query) , and the object is selected from a list. These query results may be continuously deleted and added to the library. Fields include: 'Uitschrijver', 'Verzekerde', 'Prestatie', and 'Uitvoerder'. Those fields should occur in the invoice.

3. **Syntax Fields:** The third type of field requires the input to be in a certain syntax. Fields are: 'Bedrag', 'Datum', 'Factuurdatum', 'Factuurnummer', 'Aantal'. Those fields should also occur in the document.

2.1.1 Multiple Lines

Some field types may occur multiple times in an invoice. This can be interpreted in two ways: (1) By considering them as separate fields: 'Prestatie1', 'Prestatie2', 'Prestatie3', 'Bedrag1', 'Bedrag2', 'Bedrag3'. Here, the relationship between fields is conserved, as 'Bedrag1' belongs to 'Prestatie1'. The disadvantage is that many more fields need to be extracted and more classifiers have to be trained. (2) You can also consider it one field: 'Prestatie'=[item1, item2, item3], 'Bedrag'=[item2, item3, item1]. This would require that you make some assumptions to connect the 'Performance' and 'Bedrag' together. You could assume that they are on the same line (y-coordinate). The order in which they are obtained is probably the correct one but this is not certain. One mistake can potentially deliver multiple faulty results.

The first approach requires only the training of extra classifiers and all implementations from the literature can be applied. There is little to no research done on the second approach. Some implementations from literature can only predict one item. For this research, we use the first interpretation of the problem as the implementation will be simpler. Using the second approach would cause more time-consuming programming which is not preferred for this general research.

2.2 Approaching the Problem

Category field inputs are already given by the source of the invoice or can easily be inferred from other input fields. This report focuses only on filling the autocomplete and syntax fields.

2.2.1 Approaching Autocomplete Fields

Autocomplete fields require a choice from items in a library. It would be possible to **'search for keywords'** in the invoice. Once an invoice keyword is found that also occurs in the ForceZBO library, the object in the library is used as an input for the corresponding field. The advantage is that the system generates all labels from the library, which is our main goal. There are two main disadvantages: (1) It could occur that the method returns multiple labels for one field, and there is no distinction between different options in terms of correctness. (2) Also, an invoice contains 200-1200 possible combinations of words. Sending 200-1200 requests to the server for at least four fields could overload the server.

We could approach this task as a machine learning task, which means the system itself must find a way to **'classify the fields'** with the correct label by taking into account the invoice as an input. The system can choose between all the choices available in the library. Instead of presenting only one word like in the previous example, the whole invoice is taken into account, which can help in choosing the correct option. There are a few reasons why this approach would not work. (1) A machine learning solution normally incorporates only a few possible labels (approximately up to 1.000 or 10.000) and each possible label must contain at least a few sample invoices for that specific label. That means that if we take a list of 100.000 items in our library to choose from, each performance must at least occur a few times in the invoice collection, as this is a requirement for a machine learning approach. The data collection that is needed would become too large. Also, it is impossible to obtain an invoice collection with all labels included at least once. (2) The list of options in the ForceZBO library is subject to change, meaning that labels may be added (and deleted). Every time this occurs, the machine learning system must be updated to incorporate the new options. This updating process takes quite some time. It seems machine learning is not applicable in this manner.

Instead of using all the possible options from the ForceZBO library as different labels, we use the autocomplete field names (e.g. 'Uitschrijver', 'Verzekerde', 'Prestatie', and 'Uitvoerder') as labels and try to **'classify the invoice words'**. Here we see that the list of possible labels becomes very small and every label occurs probably at least once per invoice. This is called the 'named-entity recognition'task (in literature) which we will also refer to as a word classification task, as you are labelling some/all of the words in the invoice with a specific label. After the machine learning task classifies word(s) from the invoice, a server request may retrieve the corresponding library label for those words. The server will not

be overloaded. Field inputs may be generated that were never seen before. Most research applies such a word classification method to extract information from invoices.

2.2.2 Approaching Syntax Fields

A server request is needed to retrieve the correct label for the autocomplete fields, but for the syntax fields, no such request has to be made and the output just has to obey certain rules. The autocomplete fields approach is applicable here according to the same argumentation. In this case 'Bedrag', 'Datum', 'Factuurdatum', 'Factuurnummer', 'Aantal' should be the options to label the words in the invoice. Afterwards the string must be formatted to the correct structure, because the syntax from the extracted word and the syntax required can deviate slightly. E.g. *11-11-1994* to *nov. 11th '94*, *€1,394.00* and *€1.394,00*

2.3 Machine Learning Solution

The previous chapter proposed a classification system that can generate predictions. However, some prior steps are needed to create such a system, see figure 2. Before any deployment can take place, a model must be learnt and optimized, which can be used for predictions afterwards. The 'model building' is a manual process that includes the following steps:

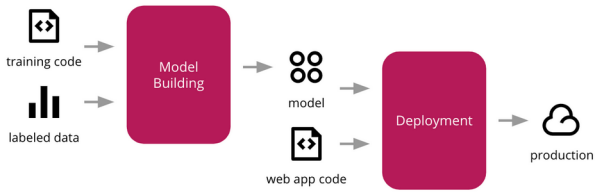

Figure 2: high-level structure of the machine learning process. Source: (Sato, n.d.)

- To train a model, a data collection must be obtained. In Topicus' situation, data is generated by the end-users. A data-set which is similar can also be used but this may hurt the performance when it is deployed in an environment with different invoices. Because of extremely limited data availability on the web, and because none of those available invoices include the fields that are expected to occur in ForceZBO invoices, it is a necessity to access this user-generated data.
- Pre-processing: invoices are semi-structured documents and the way the input is presented to the model can influence the performance. Determine the way the invoice is presented to the model. This will be described in paragraph 2.4.
- Determine the learning algorithm (model). What does the model look like internally? This will be described in paragraph 2.5.
- Optimize the model over the available data.

- Determine the way you quantify results to see how it performs.

Steps may be repeated on new data, a bigger data-set, or a different model, to improve performance again. The pre-processing steps and the optimized learning algorithm are then carried over to be deployed in the system. Below we analyze some methods that are used in pre-processing and in the learning algorithm.

2.4 Pre-processing Techniques

To apply classification on the words of the invoice, some processing of the invoice must be applied. Below are the two main techniques which are both necessary pre-processing steps in state-of-the-art solutions. Here we refer to solutions provided by existing literature.

2.4.1 Optical Character Recognition (OCR)

As we saw in section 2.1, we need to label some words in the invoice, which means that knowledge about which words are where in the invoice is important. Optical Character Recognition (OCR) is the process of converting images containing typed or written text into encoded text. Recent OCR systems like Tesseract and OCRopus use neural networks (explained later) and were trained on big amounts of documents for the optimization of this network. Those systems recognize not only characters but also words, sentences and extract their locations in the text. Section 8 gives some extra recommendations on this topic.

2.4.2 Word Embedding

After we extracted the words from the invoice, we can use a (pre-trained) word embedding. Word embeddings include a specific numerical representation for every word. This representation is different for every word but similar for semantically similar words. Example representations are given in figure 3. In essence, a word embedding is a vector in a multidimensional space. Most of the word embeddings are generated using neural networks, which we talk about in the next section. Before we have seen any invoice, the system can already extract extra semantical information from the invoice by applying word embeddings. Examples of word embeddings are GloVe (Pennington, Socher, & Manning, 2014), Word2Vec (Mikolov, Chen, Corrado, &

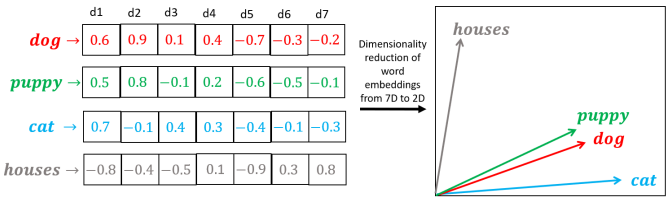

Figure 3: example of the embedding for 4 words based on its semantic meaning. The plot is shown in 2d but a 7d plot would be needed to draw the vectors. Source: (Rozado, 2020)

Dean, 2013), and BERT (Devlin, Chang, Lee, & Toutanova, 2018) that are already learnt and can be implemented at the start of a model. We will be using BERT in our experiment as it is the optimal choice in many similar problems too and is also used in invoice extraction solutions in literature. There is also a multilingual and a Dutch version available. Many embeddings are open-source.

As an example of how embeddings are advantageous in our case: when the language of an invoice is changed, the word embedding generates almost the same representation, because semantically the invoice is the same. When different prices/amounts occur, the representation also stays stable, as amounts share a similar vector representation. In all cases, the correct predictions are generated, while the input can be very different in terms of characters and words.

2.5 Learning Algorithm

After pre-processing, a learning algorithm can convert the invoice to a prediction. A neural network is just one of the machine learning methods available, but due to its achievements on invoice extraction this will be the only method we will discuss in this report. This is the part in the pipeline that is trained/optimized on data as described in section 2.3 and will provide a prediction.

2.5.1 Neural Networks

Neural networks (NN) simulate the dense connections of the brain, making it able to learn things, recognize patterns in the input, and weigh importance of certain features to finally come to an output. Neural networks are very flexible in their structure, as they can take any form the data scientist wants. The downside is that the data scientist cannot control what the model learns exactly so the model must be validated by using measurements on a test set of samples, and cannot be validated by looking at the internals of the system.

Neural networks occur in many different forms. Two specific types play a prominent role in invoice extraction. **Convolution Neural Networks** (CNN), are good at finding patterns in two-dimensional data, for example images. **Recurrent Neural Networks** (RNN), see fig 5 further on, are great for classifying one-dimensional data such as texts. We will be discussing one-dimensional and two-dimensional neural networks for word classification in section 3.2 and 3.3.

Neural networks generate an output by providing an input (image, text etc...). This generation process can be optimized by comparing it to the true label. All the variables that influence this output are slightly adjusted. A simple neural network is shown in figure 4. This is the basic building block of bigger neural networks.

In this case, the variables that can be adjusted are w_0 to w_{12287}, the input variables are x_1 to x_{12287} which can correspond to pixel values, word embedding values, other (engineered) features, or the output of another part of the network. The activation is made by a sum of the weight-input multiplications $w^T * x^{(i)}$ $(= w_0 * x_0 + w_1 * x_1 ...)$. Variables w can be adjusted so that the formula generates the required response. Afterwards, when the same sample invoice is used again, the label will be closer to the true label, increasing the performance of the network. However, the adjustment must be minimal, as you are partly overwriting previous adjustments. A data-set should be repeated multiple times and in random order so the neural network can extract the correct knowledge. Combinations of these building blocks can form bigger neural networks. For example, the output of one building block can be the input of another one.

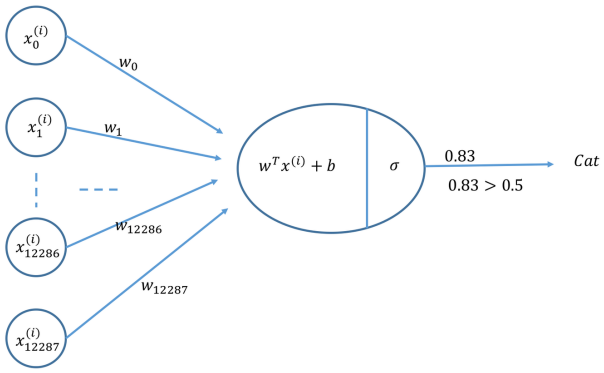

Figure 4: example of simple neural network

Almost all other language processing tasks (e.g. machine translation, speech recognition, named-entity recognition etc...) are also solved best with NN's (Otter, Medina, & Kalita, 2019). The application of neural networks has also been very effective for computer vision tasks (e.g. object detection, image classification, face recognition) (Voulodimos, Doulamis, Doulamis, & Protopapadakis, 2018).

3 Invoice Extraction Concepts

In the literature there have been numerous attempts to apply keyword extraction to invoices. The literature can be split up into three distinct approaches. (1) Non-Machine-Learning methods focusing on template matching or rule-based approaches without the use of a Machine Learning approach. (2) Machine Learning methods that were originally designed to interpret text. There has been a lot of research on the interpretation of natural human text by computers. This knowledge can be used to extract the correct keywords. In this case, spatial information is partly omitted and the text is sequentially provided to the model. (3) The most recent developments include models where the spatial information is conserved. The input consists of a two-dimensional grid filled with features. What those features are differs per method. At the end of this section we also discuss sequential learning methods.

3.1 Template Matching

Automatic Indexing (Esser, Schuster, Muthmann, Berger, & Schill, 2012) and **Intellix** (Schuster et al., 2013), are non-machine-learning template matching methods and they both globally share the same approach, using a database of (1) PDF correspondences and (2) spatial locations and value of the key fields in the document. A k-nearest-neighbour (k-NN) classification is applied to find similar documents (and templates) matching the input PDF. k-NN is a classification method that tries to find a set of invoices that share the most similar features. When some matching invoice(s) are found in the database, it tries to extract the fields from the new invoice by using the locations from the invoice(s) that are similar. The features can be manually defined. In their implementation the k-NN classifier uses two types of features; graphical and semantic similarity. Graphical similarity, similar to the methods proposed by Hu et al.(Hu, Kashi, & Wilfong, 2000), matches the PDF's by dividing it over a grid, and classifying every cell as empty (0) or non-empty (1) based on the amount of non-white pixel values in that cell. The resulting binary values are then compared for similarity. For semantical similarity, a grid is filled with words from the corresponding location in the invoice and a comparison is made similar to the graphical similarity comparison.

The template-matching approach has three disadvantages. (1) If there is no template match or the template has never been seen, the chances of extracting the right words are zero. (2) Both methods are not able to handle different lighting/photo conditions and label locations in the invoice and maps the templates 1 on 1. No distortions are permitted. (3) When a new template is recognized, the locations of the key fields in the invoice have to be manually annotated.

Dhakal et al. (Dhakal, Munikar, & Dahal, 2019) created a similar template-matching method very recently, and used a mathematical method called Singular Value Decomposition to make it resistant to different lighting conditions and image dimensions. The key field locations however remain static and cannot diverge from the template locations when the image does not correctly align. We would like to find methods that are resistant to those kinds of augmentations.

A **Rule-based approach** may provide us an easy solution to the problem. We omit any machine learning methods and set fixed rules on the features of a label. For example, a date string has a distinct layout. However, when multiple dates pop up in the invoice, you cannot extract the right date with this rule-based approach. Also, some labels may share the same features/layout, such as item pricing. It would be possible to generate extra rules, as the total price is at the bottom and the first price probably corresponds to the first item. This probably increases the performance on the data. By manually applying this "feature engineering", the optimization for a very large set of invoices takes an enormous amount of time. That is why this kind of optimization is not feasible for large sets of different invoices.

One-shot methods refer to learning when a sample may only be used once. Many template matching methods are able to learn in this fashion. Machine learning methods that use such a learning metaphor is not effectively possible. Because those methods rely on randomization and the availability of many invoices at the same time.

3.2 1D Understanding

Normally, text is provided on a two-dimensional screen or paper, while human reading is applied in a sequential one-dimensional way. The meaning of the text does not change when it is displayed on one line. Word classification can be applied to such a text to extract or label the required words. An invoice can also be seen as a sequentialized text. An advantage is that many solutions for natural text can be applied on invoices. A disadvantage is that the two-dimensional structure in the invoice is lost due to this conversion. This word classification on one-dimensional text is also called Named-entity Recognition (NER). An example can be seen in figure 5. The image clearly shows that the input must be represented in a sequential order. The output can classify all the words separately in its corresponding fields. The inner workings consist of the building blocks described in section 2.5.1

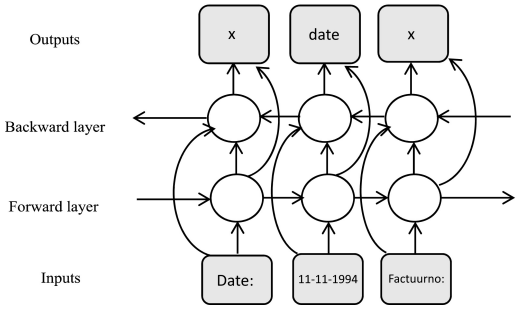

Figure 5: example of recurrent neural network

CloudScan (Palm, Winther, & Laws, 2017), by Palm et al., was the first sequential neural network implementation that was specifically created for invoice extraction. The input included word embeddings and some other features. It performance on seen and unseen templates was already quite high, but we have to take into account here that the documents are neatly scanned. The input consisted of engineered features along with the raw words.

3.3 2D Understanding

While most learning algorithms involve sequencing all the invoice words, newer techniques use CNN's (explained in section 2.5.1) that can extract structures from two-dimensional data. As stated in the previous section, CNN's only functions on input of constant size. Therefore we generate an empty grid for every invoice and fill it with the characters, words, word embeddings or other features in the correct grid cell at the location of the corresponding

information in the invoice. Because of the effectiveness of CNN's on images, the technique has also had its advantages on extracting information from invoices. **Chargrid** (Katti et al., 2018) and **Wordgrid/CUTIE** (Zhao, Niu, Wu, & Wang, 2019) use CNN's and show that they can outperform many sequential models, like CloudScan. Chargrid uses a grid filled with characters (words) in corresponding locations. Wordgrid/CUTIE uses a word embedding (not pre-trained). **BERTgrid** (Denk & Reisswig, 2019) performs even better and showed a consistent improvement over methods from Katti et al. and Zhao et al. by using the BERT word embedding. Figure 6 corresponds to the Wordgrid from Zhao et al. .

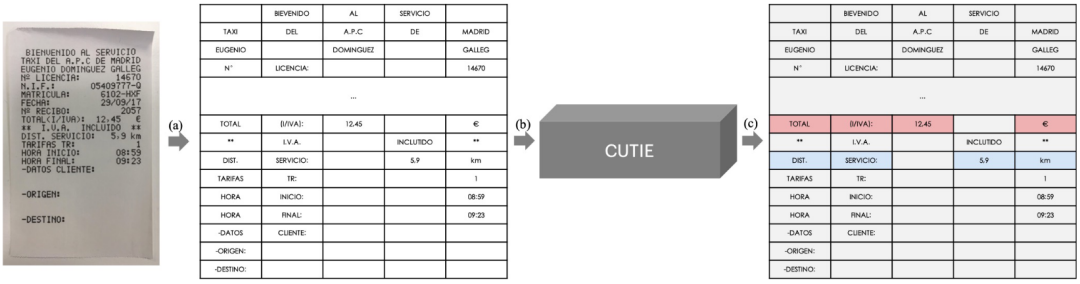

Figure 6: Example neural network for classifying each input word (embedding). Source: (Zhao et al., 2019)

The output from the network is a grid of prediction scores for every word. In the image from figure 6 it is displayed with colors, but in reality those are values between 0 and 1 indicating the confidence of belonging to a certain class (color). In the figure, the CUTIE model is predicting that some grid cells belong to the "total amount" field (red). When learning the model, we need to know which grid cells should have been predicted. In our datasets, labels consist of a string as the ground truth, which is not a grid. The authors of the paper overcome this by manually annotating the dataset, i.e. manually labelling all the words in the text, connecting them to the corresponding field. We could automate this process. This is where distant supervision techniques come in.

3.3.1 Distant Supervision

Upon closer inspection of the one- and two-dimensional learning algorithms, the data that is used to train the networks should include location data or bounding boxes. However, in ForceZBO no location data is generated, only the string labels for fields are generated by users. That means that the location data must be extracted from the invoice first. We compare the autocomplete and syntax field information that was manually annotated by users with OCR extracted text from the invoice. This way we can find the location of the words in the text so we know what words should be extracted from the text for the fields in this specific invoice. This process is sometimes called 'Distant Supervision' (Palm et al.,

2017; Palm, Laws, & Winther, 2019; Holt & Chisholm, 2018). This distant supervision method may generate other labels than when a human would generate the location data.

An ***OCR mistake*** may cause a mismatch and the words in the invoice are not recognized. This error causes the true labels to be generated incorrectly. Also the ***format*** of the keyword in the document may be different from the manual annotation causing a mismatch. When we try to learn over a value that has ***multiple occurrences*** in the document, it is impossible to know to which one it is referring to.

It is possible to (1) manually annotate the dataset so that it includes corresponding locations, or (2) use an automated matching method to connect the document words and the key field labels. In experiment 1, we introduce a distant supervision method to tackle this problem. The quality of the prediction is highly dependent on the quality of this distant supervision method.

3.3.2 End-to-End Model

While most previously mentioned approaches were characterized by the fact that word-level labels were needed (distant supervision), the "Attend, Copy, Parse" (ACP) method (Palm et al., 2019) is an exception. To achieve some understanding of this model, we need to know what N-grams are. N-grams are sequences of N words. Where N can be a number decided by the user. An example of a 3-gram is "The Three Musketeers". If you would like to extract the 2-grams, that would only be "The Three" and "Three Musketeers", but not "The Musketeers" or "The Three Musketeers". This methods extracts all the N-grams from the text up to N=4, so including 1-grams (single words), 2-grams, 3-grams and 4-grams. The list of all the n-grams in the invoice should contain the correct labels for the fields. Instead of using only words, like in previously mentioned grid models, it uses a combination of multiple "modalities", including the n-grams, the original (low quality) version of the input (pixels), word embeddings, character embeddings, patterns, and specified locations of the words in the text. The output of the model is a prediction over all the n-grams in the text. It is similar to the output in figure 6 but not for single words but one prediction for combinations of words.

The important difference is that the complete pipeline is mathematically differentiable and thus end-to-end optimizable. That means, instead of having to generate the true word-level labels before learning, it can use the true field values directly as labels to optimize the model.

If some n-gram in the invoice is discovered to have a certain template, the model corrects the template in the invoice to the required template for the extraction. One limitation however is that in the learning phase, you need to provide not only the correct template but also provide a list of possible templates in the invoice. In most cases this kind of parsing is needed for dates and prices, as they can come in different forms. By providing all possible

templates, an extra neural network can be learnt to do this pattern conversion itself. It learns how to move the characters and symbols in a string to change the template, e.g. *"11-11-1994" -> "nov. 11 '94"*.

The results seem very promising, as it outperforms the older CloudScan model (Palm et al., 2017) by increasing the average accuracy from 0.75 to 0.83. Their dataset (1.1 million invoices) consisted of digital pdf's but also 37% scanned paper invoices. When testing their system, they used invoice templates that were not yet seen by the system. They also predict that the maximum possible score on their dataset is 0.84 due to (1) OCR mistakes and (2) by the manual labelling words that are not present in the document. For example, manual annotators could be correcting a mistake in the invoice. This recent paper is very comprehensible and applicable in Topicus' situation. The results seem much better than other models which need distant supervision.

Although the end-to-end models are more complicated compared to the simpler distant supervision methods, Palm et al. use many modalities in their input and prove in their paper that they are close to the maximum possible score.

3.4 Lifelong Learning

In the introduction, we referred to sequential learning. This learning approach means that every invoice can only be seen once by the learning algorithm and in the order that they arrive in. Unfortunately, there is no machine learning implementation that can learn in this fashion. However, literature proposes methods that can learn over a new collection of data, while retaining the information that was previously learnt. This means that the system can be updated with new information, but it still has to obey some other rules of machine learning. The dataset has to be big and accessible in random order at one point in time. This way of updating the model is called lifelong learning, or in literature inconsistently referred to as Continual Learning, Data Stream Learning, Incremental learning, Online Learning and Out-of-Core learning. It can be described as the "The ability to continually learn over time by accommodating new knowledge while retaining previously learnt experiences" (Parisi, Kemker, Part, Kanan, & Wermter, 2019). Data may not always be available, or data may be privacy-sensitive. In this case it would be better to have a model that is less data-hungry and can retain much old knowledge from previous iterations. The main problem of lifelong learning is an inherent problem of neural networks; the forgetting of old knowledge, referred to as Catastrophic Inference. (McCloskey & Cohen, 1989). In our case, we would need lifelong learning when the size of the dataset that is available at a single point in time is too small, so that we need to remember previously learnt information when we re-learn the model over the new data. In the next sections, we propose three solutions, but separate research has to be done to see whether it is effective on the extraction of invoices. We will not be applying any of the methods due to limited time available. As no literature

talks specifically about the combination of invoice extraction and lifelong learning, this may potentially need a lot of extra research.

3.5 Solution to Catastrophic Inference

Catastrophic inference is inherently linked to deep learning: adapting is learning, but adapting is also forgetting. A neural network is not an endless database, but rather an adaptive predictor. This is still an ongoing problem in the field. Early attempts to resolve this issue consisted of systems that replay old data (Robins, 1993) (Robins, 1995) together with the new data. Those methods are still used in more recent research (Gepperth & Karaoguz, 2016; Rebuffi, Kolesnikov, Sperl, & Lampert, 2017). Another approach is an attempt to protect the already learnt weights when learning new data (Kirkpatrick et al., 2017; Zenke, Poole, & Ganguli, 2017).

3.5.1 Data repetition

Some solutions were already proposed in 1995, which include just repeating old data, to keep the model effective for not only the recent data. This method tries to stick to the original idea of traditional machine learning, to keep the dataset balanced by not only using the most recent data. (Robins, 1993)

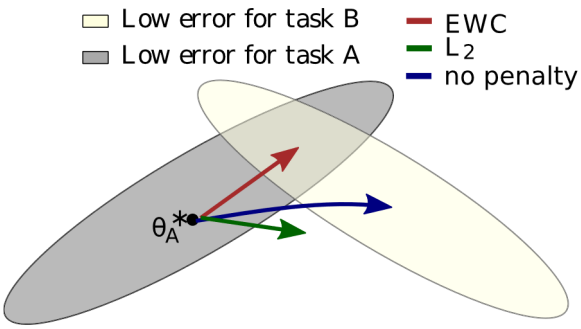

Figure 7: From Kirkpatrick et al.(Kirkpatrick et al., 2017), visualization of the workings of EWC, constraining important model weights for task A.

3.5.2 Elastic Weight Consolidation

(Kirkpatrick et al., 2017) has gotten much attention with his replacement for stochastic gradient descent (= traditional neural network optimization), called "elastic weight consolidation", i.e. EWC. This technique scores the weights with an influence score, and making them harder to learn when they are important for a previously seen task. Figure shows a graphical description of EWC. It displays the parameter space (= weights of the network) and tries to find the best combination of weights (= θ_A^*) for the new task (B). Naturally, the model will not take into account the old task and will only optimize the new task. This can be seen in figure 7. This causes performance degradation (blue arrow) for the old task A. In our case

the old task is the collection without the new templates, and the new task is learning the newly received templates.

3.5.3 Intelligent Synapses

Intelligent synapses (Zenke et al., 2017) is the same as Kirkpatrick but calculates the weight importance in a continuous fashion rather than in a certain point in time between tasks. This approach may be more useful for our scenario, as we do not have clear descriptions of tasks, and we do not know when a new task (or template) comes in.

3.5.4 Progressive Neural Networks

The progressive neural network was proposed by (Rusu et al., 2016) and implements a new column next to the original model. Instead of freezing part of the variables in the model (EWC), it is possible to add new variables to the network and the old ones are frozen. The model can be trained on the new task and it can choose between the old features and creating new features.

4 Experiments

We will be conducting two separate experiments on the subject of extracting information from invoices. The first experiment considers two custom implementations [1] of distant supervision and we compare the two by comparing the accuracy of information extraction. The second experiments applies the best implementation of the first experiment and an existing implementation[2] called "Attend, Copy, Parse" that does not need distant supervision. Both try to extract information from unseen invoices.

4.1 Optimization Pipeline

Before we can test these implementations, we need to optimize it. We use the four steps also provided in section 2.3 .

4.1.1 Choose a Representation (pre-processing)

For our custom implementation, we will be using the existing Tesseract OCR. Before we can apply OCR, we convert the PDF to an image type, like JPEG, with a pdf-to-image converter. We use a pre-trained word embedding called BERT to convert every word to a word embedding. We create an empty two-dimensional array of 100x70. This should be big enough as it is highly unlikely that a PDF displays more that 100 lines on a single page. It

[1] https://github.com/CaspervanAarle/InvoiceExtraction
[2] https://github.com/naiveHobo/InvoiceNet

is extremely unlikely that there will be more than 70 words on a single line. It is important that the grid is big enough so that all the words in the invoice can be placed in the correct location without overwriting others. The bigger the grid, the longer it takes for the model to be learnt, because computation time goes up. Every word embedding is placed in a grid cell corresponding to its location in the invoice.

4.1.2 Choose a Learning Algorithm

We used the Cutie paper (Zhao et al., 2019) as an inspiration for our neural network. We could have created a custom neural network, but it is better to use an existing structure that is proven to perform. The architecture used is summarized in table 4 in the appendix. The output of the network is a prediction for every grid cell over all the possible labels. The amount of fields depends on the problem that needs to be solved. Let's assume we have 4 different fields to fill. We will need 5 predictions for every cell, as an extra field is needed to identify the cells that do not belong to a specific field (that is the 'don't care' field). In this case the structure of the output will be 100x70x5. For 1 single cell, the 5 prediction values will sum up to 1. Those are probability values. All 5 values in the cell indicate the chance of that cell belonging to one of the 5 fields. We label the cell with the probability that is highest. One downside is that one cell can only have one label, however this will probably not cause any problems.

4.1.3 Optimize Learning Algorithm

The model (neural network) must be learnt by using a large collection of invoices. We generate the prediction fields for every cell in the grid for all the invoices in the collection by applying the learning algorithm on invoices. By comparing the prediction fields and the true fields, see section 8, we can optimize the model by applying 'stochastic gradient descent'. This method changes all the variables in the network so that the predicted label will resemble the true label a bit more. This is done for all the cells and is repeated for every invoice. This process is repeated multiple times to optimize performance. How we generate the true field labels/values is described in section 4.3.

	Example Fields			
	Company	Address	Total	Date
Predicted Field	0.02	0.10	0.87	0.01
True Field	0	0	1	0

Figure 8: Probability values for every cell that must be compared to the true field prediction

4.1.4 Quantify Results

After the learning algorithm is optimized over the collection. We need to test the performance of the learning algorithm on another set of invoices, as we need to know how it

performs on new invoices that it has never seen. This collection of invoices does include two different types of invoices. There are new invoices that use a template that was already observed previously in the learning phase. It can also include new invoices that use templates that the model is unfamiliar with. In the first experiment we measure the performance over a random subset of the dataset. The second experiment will only test on templates that have not been seen.

The performance of a model can be determined by calculating the average precision (AP). We calculate this for every field separately. When applying AP, we check whether the labels that were predicted for a specific field correspond to the true label. A correct prediction of a field is only the case when the exact cells required are given by the learning algorithm. This amount is then divided by the total amount of invoices. This is repeated for all the fields required. We use an extra metric called softAP, checking whether at least the required label were included. That means a prediction field may include 'false positives', labels that were not in the true label but were still predicted. We argue that false positives can easily be found and deleted.

4.2 Data

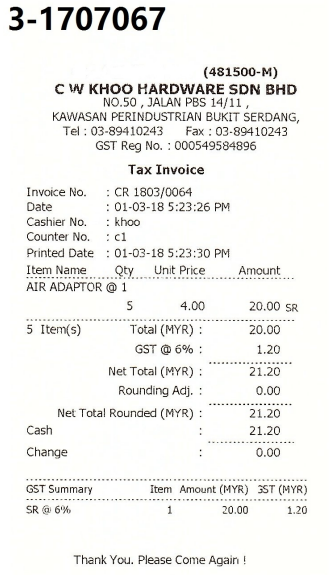

Figure 9: Sample invoice from SROIE dataset

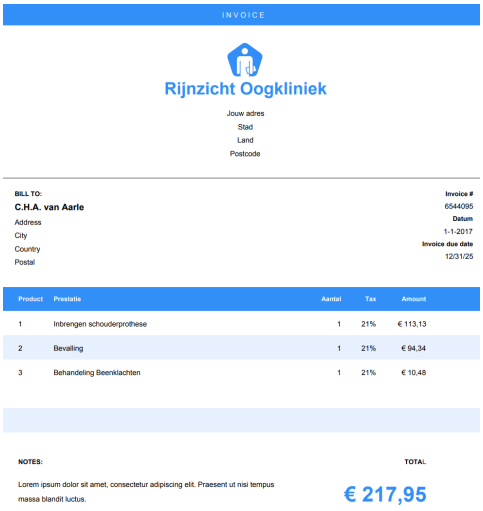

Figure 10: Sample invoice from CUSTOM dataset

Due to healthcare invoices not being accessible, it is not possible to yield a learnt model that is immediately useful for Topicus. However, testing may prove the effectiveness of the

model which can later be applied to a different collection of healthcare data. We will be conducting the experiments on two different datasets.

(1) The open-source **'SROIE dataset'** is used. This dataset is characterized by the not so neatly scanned invoices, very long field labels, and many different templates. Invoices are not healtcare related

Due to the unavailability of data on the internet, we have generated our own **'CUSTOM dataset'** (2) consisting of 1000 invoices. The prices and dates are random, the organization names and performances delivered are sampled from a limited set of names. Our dataset is characterized by the small amount of different templates, and the clear representation of the document because they were never scanned. Invoices are healthcare related. Both dataset are important to take into account, to see how collections with different characteristics perform.

Every invoice has an attached text-file that includes the field labels and label values, see 11. The labels are not yet grid labels that we need but are separate string objects. In experiment 1, we try to bridge this gap.

4.3 Experiment 1: Distant Supervision

Distant Supervision can be used in cases where the exact labels for a task are not available but can be inferred from other data. In this case we need to generate labels for every invoice word when for every field label only one label value is available in the form of a string, see figure 11. Because manual annotations of invoices are available, we can directly compare all the words in the invoice with the label value to see which words in the invoice correspond to which field. We propose two ways to do this comparison, Exact Comparison (section: 4.3.1)and Fuzzy Comparison (section: 4.3.2).

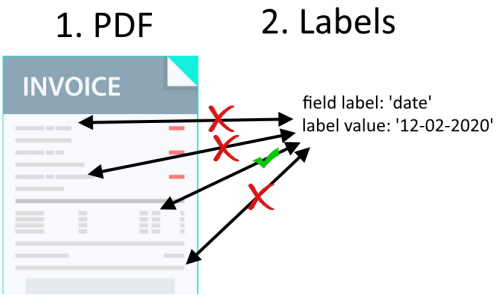

Figure 11: When invoice and labels are available, all invoice words are labelled with the corresponding field label. In this example, one word is labelled with the correct field label "date".

We use both the SROIE dataset and the CUSTOM dataset to validate our results. For the CUSTOM dataset, the training set size is 800 and the test set size is 200. For the SROIE dataset, the training set size is 500 and validation set size is 126. Training time for both methods was 4 hours on a local computer.

4.3.1 Exact Comparison

By applying an exact comparison (EC) between the words in the invoice and the manually annotated words, we derive what invoice words correspond to which field label. We label the invoice words with 'dontcare' when the word does not belong to any field label. The invoice word must match the label value exactly or it must match one of the words in the label value.

4.3.2 Fuzzy Comparison

The Fuzzy Comparison (FC) compares the individual words in the text with the label and taking into account a certain error rate. More technical, we calculate the Levenshtein distance between the words in the invoice and the field. This distance is a value taking into account the number of edits (character addition, removal, change) and the length of the strings. The simplified calculation is $100 - 100 * \frac{\#edits}{(l_1+l_2)}$. L_1 and L_2 are the lengths of both words, so longer words need more edits for the same similarity score. We hypothesize that it can overcome OCR mistakes, so that the words that were not correctly OCR extracted from the text, are still labelled with the correct field label. We hypothesize that this method will generate higher quality word labels, by taking into account spelling and OCR mistakes.

4.4 Experiment 2: Unseen Templates

A part of the invoices that will occur will have layouts that the model has never seen before. It is incorrect to assume that the performance observed in the first experiment will be similar to the performance on unseen templates in the second. With the second experiment, we attempt to compare the performance of our custom implementation with a state-of-the-art end-to-end implementation "Attend, Copy, Parse" (ACP).

We omit the parse methods from ACP for simplification. The parse method is meant for prices and dates that are in a different format, but we will be using the same format in all the templates and in the labels. Two templates from the CUSTOM dataset will not be used for this experiment, as the formats from the prices and dates in those invoices are different.

We only use our CUSTOM dataset as we can clearly split the dataset on unseen templates. We train on 7 templates and test on 1 template not included in the training set. The training set size is 700, and the test set size is 100. This test will be repeated once on a different excluded template to generate a better approximation. Due to the dataset only containing 8 different templates, it will not be sufficient to draw strong conclusions from this. However, results may still be useful. Only the AP metric is used, not softAP because 'Attend, Copy, Parse' always predicts one label string, not multiple words/strings. Training time for both methods was 4 hours on a local computer.

5 Results

We will mainly interpret the AP scores, not the softAP scores. softAP may potentially include many false positives and it is not a good measurement. However, one may find it useful to see post-processing potential with this softAP score. In the tables in the next sections, the different rows indicate the different fields. A column indicates a specific machine learning solution.

5.1 Experiment 1

SROIE class	EC AP / softAP	FC AP / softAP
company[1]	**0.206** / 0.206	0.175 / 0.254
date	**0.524** / 0.619	0.397 / 0.397
address[1]	**0.063** / 0.079	0.016 / 0.063
total	0.206 / 0.413	**0.222** / 0.317
avg.	0.250 / 0.329	0.202 / 0.258

Table 1: EC and FC scores on the SROIE test set for known templates

CUSTOM class	EC AP / softAP	FC AP / softAP
factuurnummer	**0.970** / 0.970	0.940 / 0.940
factuurdatum[2]	**0.730** / 0.730	0.720 / 0.720
uitschrijver	0.940 / 0.950	0.940 / 0.940
bedrag[2]	**0.750** / 0.760	0.700 / 0.700
item1-naam	**0.930** / 0.990	0.890 / 0.910
item1-prijs	**0.830** / 0.860	0.820 / 0.820
item2-naam	0.950 / 0.990	**0.970** / 0.980
item2-prijs	**0.950** / 0.960	0.870 / 0.880
item3-naam	0.950 / 0.950	0.950 / 0.950
item3-prijs	0.960 / 0.970	0.960 / 0.970
avg.	0.896 / 0.913	0.876 / 0.881

Table 2: EC and FC scores on the CUSTOM test set for known templates

Both datasets are very distinct, and this is reflected in the precision scores in table 2 and 1. On average, the amount of mistakes in the SROIE dataset is 7 times higher. We hypothesize that this is caused by different aspects of the SROIE dataset. First, this dataset is 30% smaller and this potentially influences the usefulness of this dataset. Second, the dataset is not neatly scanned like the CUSTOM dataset. This may be the cause of extra OCR mistakes and may cause misinterpretations with word embeddings, and the learning phase is very dependent on a comparison between the raw words in the text and the label string.

Some fields have a ([1]), indicating that the labels consist of many words (up to 12). The prediction is only considered correct when all the words are predicted. This evaluation therefore penalizes long sentences. The Date and Total fields are more representative fields for our research, as ForceZBO will not have to extract extremely long sentences.

([2]) 'Factuurdatum' and 'Bedrag' perform not as well as the other classes. This is caused by some invoices that formatted the total price and date differently, i.e. *"27th feb 2020"* instead of *"27-02-2020"*. Some instances of 'bedrag' had a non-european format, and were not recognized as being the same value as in the corresponding manually annotated field. Due to our simple comparison methods, those words are not recognized as belonging to that class, and are never included in the learning phase. That in turn causes the model to also not predict them at prediction phase. This shortcoming can easily be overcome by parsing the string in different formats before the distant supervision comparison. When we introduce such a parsing system, we expect the AP scores to match the other fields better, such as 'uitschrijver' (0.94), factuurnummer (0.970).

5.2 Experiment 2

Model	CUTIE + EC	ACP	CUTIE + EC	ACP
Learned templates:	2,3,4,5,6,7,8	2,3,4,5,6,7,8	1,3,4,5,6,7,8	1,3,4,5,6,7,8
Tested template:	1	1	2	2
metric	AP / softAP	AP / softAP	AP / softAP	AP / softAP
factuurnummer	0.595 / 0.802	**1.000** / -	0.000 / 0.000	**1.000** / -
factuurdatum	0.801 / 0.801	**0.991** / -	0.000 / 0.000	**1.000** / -
uitschrijver[3]	0.000 / 0.000	0.000 / -	0.000 / 0.000	0.000 / -
bedrag	0.345 / 0.491	**0.578** / -	0.057 / 0.136	**0.432** / -
item1-naam[4]	0.000 / 0.017	- - - - / -	0.045 / 0.511	- - - - / -
item1-prijs	0.302 / 0.353	**0.974** / -	0.000 / 0.000	**0.011** / -
item2-naam[4,5]	0.095 / 0.310	- - - - / -	0.148 / 0.227	- - - - / -
item2-prijs[5]	0.586 / 0.672	- - - - / -	0.239 / 0.239	- - - - / -
item3-naam[4,5]	0.328 / 0.647	- - - - / -	0.477 / 0.682	- - - - / -
item3-prijs[5]	0.621 / 0.621	- - - - / -	0.295 / 0.682	- - - - / -
avg.	0.368 / 0.472	0.708 / -	0.126 / 0.248	0.4886 / -

Table 3: scores for the self-built and Attend, Copy, Parse (ACP) method on unseen invoices from the CUSTOM dataset

Table 3 shows the results on the second experiment. Tests on template 1 and 2 differ in accuracy for our custom implementation. The only fields that have a high score are the 'factuurnummer' and 'factuurdatum'. ([3]) The 'uitschrijver' is not known as this company name never occurred in the learning data. ([4]) The implementation was not able to handle fields with more than 4 words. However, we included longer fields in our data collection. The model can potentially be adjusted for this. ([5]) The ACP method that we used was

unable to predict fields that do not always occur. Why this is the case is unknown, and according to the paper, this should be possible. The method should be able to handle those fields. We decided not to incorporate them in the table, as this would give a skewed view of the potential results in the future.

Some results were not included because the current ACP implementation was not able to predict those fields. Predictions with a maximum length of 4 words can be predicted, however our invoice dataset included longer n-grams. This should be fixable in future research. Precision scores were left out because of the performance was extremely bad and were not in proportion to the expected results. This is probably due to an implementation error. Future research should add extra performance scores to the missing fields. Performance on the fields that always occur is higher for the ACP implementation.

6 Discussion

The experiments has considered the following features:

- Being able to predict fields from invoice templates with a self-built method
- Being able to predict fields from unseen templates with a custom and existing implementation

Invoice extraction is a field of research that is still work in progress in recent years and even recent months. During the literature research we have not come across a solution that checks all our requirements. Most one-shot learning methods can handle situations where samples may only be shown to the system once. Unfortunately, they are unable to handle the variation in input and miss the robustness of machine learning techniques. ML-techniques however need loads of data, which is an ongoing problem in the research on information extraction due to the privacy-sensitive data that is not readily available in big amounts. We tested some of those solutions and have found that they are promising for situations where there is access to a data collection.

We tried to overcome the missing word-level labels with quite effective distant supervision methods in experiment 1. Exact matching as a distant supervision tool gives an overall better result and will be more useful when predicting templates that are already seen. The method does not provide a perfect score. This is probably due to circumstances and more research on the ACP method is needed to see its real-life performance. Longer training time, better OCR methods, and other optimization techniques may bump most scores to a perfect one.

The results from experiment 2 showed a clear under-performance of the self-built implementation with distant supervision compared to a recently released end-to-end model.

With the latter, we saw that some results were perfect after only seeing 7 different templates. Those templates (100 samples per template) were shown to the system in random order for a period with many repetitions. The results from (Palm et al., 2019) seem extremely promising in this regard. The results from the end-to-end method are most likely also applicable to templates that have been seen by the model.

7 Conclusion

The report provides an overview of machine learning techniques required to understand invoice extraction. It also mentions the most promising research on the subject of invoice extraction. We gave a brief description of the possibility of sequential learning, but concluded that that is not possible. Lifelong learning methods have some of the characteristics of sequential learning but more research is needed on this topic. We were able to build a system from scratch that could do this invoice extraction task. By giving empirical evidence, we conclude that Exact Comparison outperforms Fuzzy Comparison and is better suited for a Distant Supervision tool. However, the model does not perform well on unseen templates. This is where the Attend, Copy, Parse method excels, even after seeing only a few templates. The proof given by this report is not strong enough to conclude that ACP performs extremely well, but chances are high that it will. Unfortunately this model is not suited for extracting multiple objects of a single field. This is not necessarily an obstacle for the ForceZBO system. Building such an effective model ourselves takes more time-resources than we had available.

8 Recommendations

Invoice extraction is certainly possible, however there are some requirements for this. Firstly, access to a large collection of invoices is needed. It is not possible to find an open-source collection. ML-solutions rely heavily on this fact, and non-ML solutions will be quite ineffective on not-so-neatly-scanned documents. Implementations for unseen templates will need a follow-up research but I still see much potential in the very short term. Secondly, follow-up research is needed on the potential OCR bottleneck.

8.1 Data Requirement

It is incredibly hard to find open-source datasets to test the methods on. I would advice to request a sample of the data from the client of 1.000/10.000 invoices and their corresponding fields. Access by the user is not needed, but the model must able to freely access this collection. This gives a better idea what the performance could be, as a substantial part of the errors come from the data collection itself. This way, the model can be fine-tuned on the

real problem we want it to solve. Potentially, the resulting model from this dataset could already be quite useful in practice. It is best to train the model on as much documents as possible. Our research shows that it could potentially perform on smaller sets of data. This inevitably will hurt performance. How the accuracy increases relative to the amount of invoices is not clear and must be measured on specific ForceZBO healthcare invoice.

8.2 Extra Research on OCR Accuracy

Although the open-source Tesseract OCR Engine was lagging behind its commercial competitors (Smith, 2007) in the past, a more recent paper still included it in their comparison (Tafti et al., 2016). It was compared to numerous different input images and it did not under-perform to another commercial option. Although Tesseract may suffice for many problems, if one wishes for the best performance, one may consider Google Cloud OCR Engine or ABBYY Finereader, which both outperform the open-source option but are not free to use. It is important to do a test run of how well the OCR can extract the characters from the text from medical invoice in the ForceZBO system. This is heavily dependent on the quality of the scan. Accuracy on very neatly scanned documents will probably be 99,99% but this may drastically decrease when PDF quality drops. Imagine having 99% accuracy for every character. That means that on average on a 1000-character-invoice 10 mistakes occur. Such errors may influence the selection of syntax and autocomplete fields. Let's say 20% of the characters in the invoice belong to syntax or autocomplete fields that we need to extract. So on average 2 character mistakes will pop up in the extracted fields of one invoice. That potentially means that every invoice has to be manually checked and revised. In ForceZBO, the information must be extracted from the text without mistakes because a small mistake in the syntax fields may result in a different label that is still correct according to the syntax rules. A small mistake in the autocomplete fields will not result in any label. Comparing the best result from the invoice to the total library is not reasonable due to the list not being requestable directly. What the accuracy will be in the case of ForceZBO invoices cannot be known without testing. The research from Palm et al. used a data collection and 16% of the fields were unable to be extract due to OCR mistakes or manual annotation mistakes. The only way to check whether the OCR mistakes are a problem is to take a sample set of specific ForceZBO healthcare invoices.

8.3 Applying Continuous Learning

Applying lifelong learning requires more research, but that does not mean that the model cannot be updated at all. You can still re-train the model with as many samples that are available to you. Traditional learning does not memorize any of the previous invoices (i.e. it slowly forgets when a new learning phase is started), so the dataset must be big enough

to contain all the features and structures that you want it to learn. Some measurement can be made to check how many times the user had to change the fields, so that we see a performance degradation and can relearn the model at a certain point. My estimation is that less than 1000 invoices would severely hurt the performance, and optimizing the performance will probably exceed 10.000 invoices. I have no evidence for this claim, as this evidence can only be obtained by empirically testing it.

9 Future Work

9.1 Fuzz/Exact Total Comparison for Distant Supervision

In experiment 1 we used two different kinds of distant supervision. However, there are many ways to bridge this gap. It is possible to use the context of each word to generate a better grid label. The words in a multi-word field should only be recognized in the invoice when they occur in that same order. In contrast, exact/fuzzy comparison detected words in a multi-word field separately in the invoice. A regular expression can be used to compare to the invoice. The invoice text is then sequentialized (2d to 1d) before this comparison. When a match is found, the words corresponding to this match are labelled with the corresponding category. This method guarantees that words are labelled in the right context. For example 'and' could be a word occurring at multiple spots, however, we only want the word in the context from the complete field: 'preoperative care and surgery'. Due to the difficulty of implementation and limited need for this option, we omitted this implementation. This method potentially improves the performance for models that need distant supervision.

9.2 Multiple Lines

Literature gives little evidence on the approach of multiple lines. It is a very interesting topic to further investigate. While the methods described in this report are able to handle multiple lines, there may be better solutions.

9.3 Formatting Fields

Some labels are in a different format than provided in the invoice. This is important when applying distant supervision methods. It requires research on the invoices that will be received by the ForceZBO system. The most likely field that comes in a different format is the date. It should be possible to apply a general date parser to parse all the dates in the invoice to the correct format.

10 Evaluation

Wat ging er goed en niet goed? Ik heb de stage over het algemeen als redelijk prettig ervaren. Af en toe waren er momenten dat ik niet zeker wist wat ik moest doen. Ik zag het soms niet zitten aangezien ik niemand had om mee samen te werken. Ik vond het erg moeilijk om mijn schrijfstijl aan te passen zodat het beter te interpreteren is voor iedereen. Alleen een literatuuronderzoek was in dit geval misschien beter geweest, aangezien de lezers nog geintroduceerd moeten worden aan het onderwerp machine learning en implementaties zijn niet direct over te hevelen naar het ForceZBO systeem vanwege andere data. Ik denk persoonlijk dat ik een goed stuk af heb geleverd in de huidige omstandigheden.

Als ik opnieuw mocht beginnen, wat zou ik dan anders doen? Het verslag moet direct al op de lezer afgestemd worden. Dat gebeurde in dit geval erg laat. En experimenten werden pas erg laat definitief, aangezien ze aan een paar voorwaarden moesten voldoen: ze moesten mij persoonlijk verder helpen, het moet een valide academisch onderzoek zijn, Topicus moet ook iets met de resultaten kunnen, en ze moeten in een korte periode haalbaar zijn. Daardoor twijfelde ik erg lang over de experimenten. Dat moet volgende keer sneller.

11 Acknowledgements

I would like to thank prof. dr. ir. D. Hiemstra for his patience with me and giving me the confidence I needed. I would also like to thank Erwin Bergervoet and Jelmer Wouters for their weekly response giving me very valuable feedback.

References

Denk, T. I., & Reisswig, C. (2019). Bertgrid: Contextualized embedding for 2d document representation and understanding. *arXiv preprint arXiv:1909.04948*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhakal, P., Munikar, M., & Dahal, B. (2019). One-shot template matching for automatic document data capture. In *2019 artificial intelligence for transforming business and society (aitb)* (Vol. 1, pp. 1–6).

Esser, D., Schuster, D., Muthmann, K., Berger, M., & Schill, A. (2012). Automatic indexing of scanned documents: a layout-based approach. In *Document recognition and retrieval xix* (Vol. 8297, p. 82970H).

Gepperth, A., & Karaoguz, C. (2016). A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, *8*(5), 924–934.

Holt, X., & Chisholm, A. (2018). Extracting structured data from invoices. In *Proceedings of the australasian language technology association workshop 2018* (pp. 53–59).

Hu, J., Kashi, R., & Wilfong, G. (2000). Comparison and classification of documents based on layout similarity. *Information Retrieval*, *2*(2-3), 227–243.

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

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... others (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, *114*(13), 3521–3526.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (Vol. 24, pp. 109–165). Elsevier.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2019). *A survey of the usages of deep learning in natural language processing.*

Palm, R. B., Laws, F., & Winther, O. (2019). Attend, copy, parse end-to-end information extraction from documents. In *2019 international conference on document analysis and recognition (icdar)* (pp. 329–336).

Palm, R. B., Winther, O., & Laws, F. (2017). Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th iapr international conference on document analysis and recognition (icdar)* (Vol. 1, pp. 406–413).

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2001–2010).

Robins, A. (1993). Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In *Proceedings 1993 the first new zealand international two-stream conference on artificial neural networks and expert systems* (pp. 65–68).

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, *7*(2), 123–146.

Rozado, D. (2020). Wide range screening of algorithmic bias in word embedding models using large sentiment lexicons reveals underreported bias types. *PloS one*, *15*(4), e0231189.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K.,

... Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Sato, D. (n.d.). *Continuous delivery for machine learning.* Retrieved from `https://martinfowler.com/articles/cd4ml.html`

Schuster, D., Muthmann, K., Esser, D., Schill, A., Berger, M., Weidling, C., ... Hofmeier, A. (2013). Intellix–end-user trained information extraction for document archiving. In *2013 12th international conference on document analysis and recognition* (pp. 101–105).

Smith, R. (2007). An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (icdar 2007)* (Vol. 2, pp. 629–633).

Tafti, A. P., Baghaie, A., Assefi, M., Arabnia, H. R., Yu, Z., & Peissig, P. (2016). Ocr as a service: an experimental evaluation of google docs ocr, tesseract, abbyy finereader, and transym. In *International symposium on visual computing* (pp. 735–746).

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, *2018*.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. *Proceedings of machine learning research*, *70*, 3987.

Zhao, X., Niu, E., Wu, Z., & Wang, X. (2019). Cutie: Learning to understand documents with convolutional universal text information extractor. *arXiv preprint arXiv:1903.12363*.

12 Appendix

	name	layers	channels	details
1x	embedding block	embedding layer	128	one-hot 35000
8x	conv. block	conv2d, instancenorm	256	kernel=(3,5)
1x	ASPP block	conv2d, conv2d, conv2d	3x128	dilation = (4,8,16), concat
1x	conv. block	conv2d, instancenorm	256	kernel=(1,1), concat*
1x	conv. block	conv2d, instancenorm	64	kernel=(1,1)
1x	conv. block	conv2d, instancenorm	5/11	kernel=(1,1), softmax

Table 4: convolutional neural network architecture used to predict a label for every word.