EasyChair Preprint

№ 1763

One-Shot Template Matching for Automatic Document Data Capture

Pranjal Dhakal, Manish Munikar and Bikram Dahal

October 24, 2019

One-Shot Template Matching for Automatic Document Data Capture

Pranjal Dhakal, Manish Munikar and Bikram Dahal
Docsumo
{pranjal.dhakal, manish.munikar, bikram.dahal}@docsumo.com

***Abstract*—In this paper, we propose a novel one-shot template-matching algorithm to automatically capture data from business documents with an aim to minimize manual data entry. Given one annotated document, our algorithm can automatically extract similar data from other documents having the same format. Based on a set of engineered visual and textual features, our method is invariant to changes in position and value. Experiments on a dataset of 595 real invoices demonstrate 86.4% accuracy.**

***Index Terms*—document processing, automatic data capture, template matching, one-shot learning**

I. Introduction

Every business needs to process a lot of documents such as invoices, bills, statements, forms, etc. saved in unstructured formats such as PDF or scanned images into their accounting software. The larger ones have to process many thousands of documents per month. There are few ways to do this currently: (a) manual data entry and processing, (b) template-based extraction model, or (c) template-less machine learning approach. Manual data entry is not only time-consuming and expensive but very error-prone as well. Template-based approach requires an initial setup of hard-coded rules for every template, but it still fails badly when an unseen template is encountered [1]. Template-less machine learning method tries to learn generic features of various fields to extract so that they work well in various templates but they need to be trained with a large number of annotated documents to perform well.

In this paper, we propose a novel one-shot template matching algorithm that brings the best of both worlds—template-based engine and template-less machine learning. Our algorithm doesn't require any initial template setup, nor does it need a very large amount of data to get high accuracy. Once provided with one annotated document, future documents in the same format are processed automatically with 90% accuracy. We exploit the fact that for a specific vendor and document type, the document format is very similar, i.e., the position of annotated values and the neighboring keywords don't change much. Moreover, if it extracts a field incorrectly, the user can correct it very easily using our convenient review tool and subsequent documents in that format will learn the corrections as well.

Our algorithm saves the contextual features of every annotated value that includes information about both visual as well as textual features of not just the actual value but the surrounding keywords as well, which is explained in detail in Section III. Our algorithm also automatically finds out whether a new document belongs to any of the previously saved formats. To match a new document with a saved template, we use a combination of image similarity [2] and textual similarity [3] metrics.

The rest of the paper is organized into four sections. In Section II, we revisit the various previous approaches to solve similar problems, and also mention how our approach stands out. In Section III, we explain our algorithm in detail. We discuss the experiments and their results in Section IV. Finally, in Section V, we provide concluding remarks and possible future works.

II. Related Work

Deciding whether two documents are of the same format requires a combination of image similarity and text similarity metrics. A number of perceptual hashing methods [4], [5] have been used to detect near-duplicate images. Zeng et al. used eigenvalue matrix computed by Singular Value Decomposition (SVD) of image as features and computed similarity by comparing the angle between the eigenvalue matrices mapped into vector space [2]. Similarly, the most common approach for measuring textual similarity is the Levenshtein edit distance [3].

Flexible template-based extraction systems [1], [6]–[8] locate the required text in the document by using the distance and direction from important surrounding keywords such as field labels. Cesarini et al. [6] only look at the nearest keyword whereas d'Andecy et al. [7] computes distances and angles from every other word in the document and predicts the final location by averaging over the distances and angles from all words weighted by their *itf-df* scores.

The first serious attempt at solving the automatic data capture problem using machine learning was made by Rossum [9], [10]. Trying to mimic human brain, they process documents in three stages: skim-reading, data localization, and precise reading. Holt et al. [11] and Palm et al. [12] used a content-based template-less machine learning approach that can classify any text block into one of predefined labels thereby claiming to work in unseen document formats as well. In [11], the authors reported 92.8% percent accuracy after training the model with 300,000 documents. Completely vision object-detection models such as Faster-RCNN [13] and YOLO [14], [15], being trained on natural scenes, produce mixed results on document images. All these approaches require a large volume of annotated documents to train well.

Our method automatically utilizes template-features without needing any template-based rules. Since existing methods are either rigidly template-dependent or template-less, we cannot compare our work directly with any of them.

III. Methodology

Fig 1 shows the high-level architecture of our model. There are three major steps: template matching, region proposal, and final area selection. These are explained in detail shortly. Our model maintains a database of unique annotated templates, takes a new document as input, and predicts the annotation for the new document if such a template exists in our database.

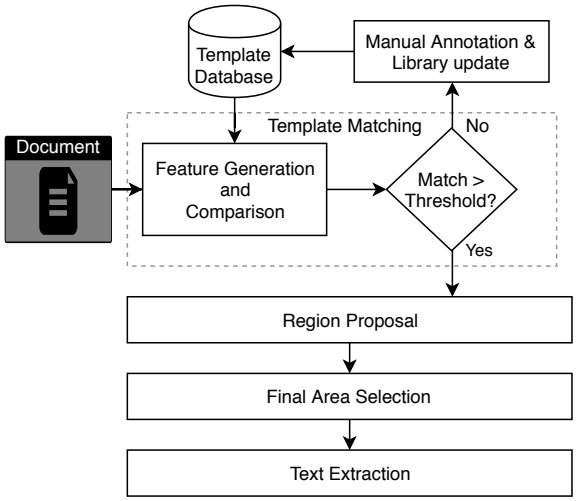

Fig. 1. Model architecture (placeholder image)

A. Optical Character Recognition (OCR)

A new document may be an image, without any texts embedded. In that case, we can get the texts present in the document by OCR. For this research, OCR can be thought of as a black box that takes an image as input and outputs a list of words and their positions (bounding boxes). There are many high-quality commercial OCR engines such as Google Vision [16], Microsoft Azure Vision [17], Amazon Textract [18]. We used Amazon Textract for this research because they produced the best result as they were trained exclusively for document images.

B. Template Matching

In this step, a matching template from the database is chosen for the input document. If there is no match, the algorithm halts and the document is sent for manual annotation. We use a combination of visual and textual similarity measures to find a match. For image similarity, we compute the SVD of images and measure the cosine similarity between the Σ diagonal matrices [2]. Equation (1) shows the SVD of a matrix I. However, before SVD, we perform some image preprocessing to adapt this metric to document images and make it invariant to image dimensions and lighting conditions.

Let I = preprocessed input document image, and T = preprocessed template document image.

$$(U_I, \Sigma_I, V_I) = \text{SVD}(I) \\ (U_T, \Sigma_T, V_T) = \text{SVD}(T) \tag{1}$$

Now, the visual similarity Sim_{visual} is given by:

$$\text{Sim}_{visual} = \cos\theta = \frac{\Sigma_I \cdot \Sigma_T}{|\Sigma_I||\Sigma_T|} \in [-1, 1] \tag{2}$$

For text similarity, we compute the fuzzy match [19] (based on Levenshtein distance [3]) of the top-n and bottom-n lines of text in the documents. Again, before the fuzzy matching, we perform some preprocessing steps to normalize the text.

$$\text{Sim}_{text} = \text{FUZZY-MATCH}(t_I, t_T) \in [0, 1] \tag{3}$$

where, t_I is the concatenation of the preprocessed top-n and bottom-n lines of text in the input document, and t_T is the same for the template document. The combined similarity is simply the sum of visual and textual similarities:

$$\text{Sim}_{combined} = \text{Sim}_{text} + \text{Sim}_{visual} \tag{4}$$

The template having the highest $\text{Sim}_{combined}$, with $\text{Sim}_{text} \geq C$, is selected as the final template for the input image, where C is a manually set threshold. If $\text{Sim}_{text} < C$ for all templates in the database, then the document is manually annotated and added to the database as a new template.

C. Region Proposal

Once the template document is selected, we use the template image and annotation to predict the approximate regions of all the fields in the input document. The annotation object is a JSON with field names as keys and the associated values and positions (top-left and bottom-right coordinates of the values) as the values. An example of an annotation object is given below:

```
{"invoice_no":
    {"position": [53, 671, 452, 702],
     "value": "INV1234"},
 "date":
    {"position": [50, 635, 312, 666],
     "value": "2019-08-24"},
 "seller":
    {"position": [259, 27, 464, 58],
     "value": "ABC Pvt. Ltd."},
 "buyer":
    {"position": [821, 445, 1153, 468],
     "value": "Zinc Enterprises"},
 "total":
    {"position": [48, 553, 419, 577],
     "value": "1,234.56"}}
```

Listing 1. An annotation sample. The coordinates are calculated from the top-left corner of the document.

In this illustration, the annotation is a JSON object where the keys are the fields to be captured and the values have the position information of the text as well as the actual text for the field value. The "position" parameter contains the top-left

(x_{min}, y_{min}) and bottom-right (x_{max}, y_{max}) coordinates of the rectangle surrounding the text.

Once we have the annotation of the matching template image, the following algorithm is used to get approximate region proposal for each field. We use the correlation coefficient R between the input document image and template image [20] to obtain the approximate region-proposal.

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x', y+y')) \tag{5}$$

where,

$$T'(x',y') = T(x',y') - \frac{\sum_{x'',y''} T(x'',y'')}{w \cdot h}$$

$$I'(x+x', y+y') = I(x+x', y+y') - \frac{\sum_{x'',y''} I(x+x'', y+y'')}{w \cdot h}$$

Algorithm 1 Region Proposal

Input:

- Preprocessed input document image (I).
- Preprocessed template document image (T).
- Template document annotation (A_T).

Output: Region proposals for all the fields in the input document.

Procedure:

1: $w_T \leftarrow$ WIDTH(T) {Template Image Width}
2: $h_T \leftarrow 1.414 * w_T$[1] {Template Image Height}
3: **for** each field in A_T **do**
4: Get rectangular area of the field. $(x_{min}, y_{min}, x_{max}, y_{max})$
5: Increase the area of the rectangle slightly in all directions.[2]
6: Crop out the new rectangular area in the template image.
7: Find the area in the input image where the cropped area from template image is most likely to match using (5).
8: **end for**

Algorithm 1 presents the pseudocode of our region proposal algorithm.

D. Final Area Selection

Next, we pinpoint the location of the annotation values in the input document by finding common words—present in both the input region and the template region of the field—and projecting the distance and displacement from the template region to the input region. This method was first devised in [1], but they looked for common words in whole document. We only look for common keywords inside the proposed regions for computational efficiency. Algorithm 2 shows the pseudocode for this algorithm.

Algorithm 2 Final Area Selection

Input:

- Input document OCR
- Template document OCR
- Region Proposals RP_I in input document (from Algorithm 1)
- Input document and Template document dimensions.

Output: Final bounding-boxes for all fields in the input document.

Procedure:

1: **for** each field in RP_I **do**
2: Find the texts that are common in this proposed area in the input image and corresponding area in the template image.
3: **if** matches > 0 **then**
4: Find the text that is closest to the actual value for the field in the template image.
5: Get the vector from the center of the closest text and the actual value for the field in template image.
6: Normalize the vector with the dimensions of template image. De-normalize using the input image dimensions.
7: Using the vector in the input image, predict the center where the value of the field is present.
8: Using this center coordinates and the dimensions of rectangle surrounding the value of the field in the template image, obtain the rectangle in the input image using appropriate scaling.
9: **else**
10: Using the dimensions of rectangle surrounding the value of the field in the template image, obtain a rectangle at the center of the approximate area.
11: **end if**
12: **end for**

E. Text extraction

Finally, once we have the final bounding box of the value, we can extract the text from the OCR data. We extract all the words in the OCR data whose area overlaps with the proposed bounding box by more than a preset threshold and then combine them in natural reading order to get the final text for each of the fields.

IV. Experiment and Result

A. Dataset

There are no publicly available dataset of modern business documents such as invoices, bank statements or employee forms, which is understandable given their strict confidentiality. Therefore, for this research, we acquired a dataset of 595 annotated invoices from a large international invoice financing

[1]This is done to make the regions have the same aspect ratio to handle documents with different aspect ratios.

[2]We expand the area in order to include some keywords common in both the template and the input image. Those keywords will help us accurately pinpoint the location of field values in the input regions proposed.

company. All of them were in the English language and there were about 35 unique formats or templates. For fields to extract, we considered the most common ones: (a) invoice number, (b) date, (c) seller, (d) buyer, and (e) total due. We used one sample each of every template as the training set and the rest 560 documents as the test set. Again, due to confidential reasons, we cannot make the dataset public.

B. Evaluation Metrics

The ground truth values don't have positional values, so we can't compute the quality of output bounding boxes. Therefore, we evaluate our model by comparing the output values of the extracted fields. Since text output may have few erroneous characters, mostly due to OCR error, we define two metrics for evaluation—MEAN-FUZZY-MATCH and ACCURACY—as follows:

$$\text{MEAN-FUZZY-MATCH} = \frac{\sum_{i=1}^{N} \text{FUZZY-MATCH}(\hat{y}_i, y_i)}{N} \quad (6)$$

$$\text{ACCURACY} = \frac{\text{no. of samples where } \hat{y}_i = y_i}{N} \quad (7)$$

where, $\hat{y}$ and y are the output and ground truth values respectively both with length N, and FUZZY-MATCH $\in [0, 1]$ is the fuzzy text matching function based on Levenshtein distance. In our implementation, we used the `fuzzywuzzy` library [19] for this.

The ACCURACY, which checks for exact match between the predicted value and the ground-truth, is affected by minor OCR errors (such as recognizing "0" as "O"). We include the MEAN-FUZZY-MATCH metric to see how our model would perform in cases where exact match isn't required.

C. Result

The results of our model on the 560 test invoices are shown in Table I. We can see that MEAN-FUZZY-MATCH is significantly greater than ACCURACY, implying that our model can leverage better accuracy if the minor OCR errors are corrected by post-processing. For instance, the buyer and seller names can be matched with a lookup table. Similarly, dates and amounts can be canonicalized to eliminate format discrepancies. Other texts can be normalized by trimming whitespaces, converting to lowercase, and so on.

TABLE I
THE PERFORMANCE OF OUR MODEL.

Field	ACCURACY	MEAN-FUZZY-MATCH
Invoice number	79.2	80.7
Date	86.4	89.4
Seller	91.5	93.8
Buyer	90.2	94.1
Total due	84.7	88.2
Overall	**86.4**	**89.2**

Considering the fact that our model doesn't require per-template rules and requires very few training samples, combined with our easy review tool, getting over 86% accuracy can result in a significant reduction in time, cost, and effort it takes for businesses to process documents.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a new way of solving the problem of automatic data capture from documents. Requiring only one example per template, our method is very effective for dataset with recurring templates.

This research has many areas for improvement though. First of all, it can't handle multi-page documents. Future works can attempt to tackle this. In addition, our model, which right now only looks at only one saved sample to predict the outputs, can be made to predict based on all saved samples of the specific template to generalize better and improve overall accuracy. Also, further research can be done to make it work with recurring fields like table line items.

ACKNOWLEDGMENT

We would like to thank Rushabh Sheth for providing us the required funding and resources. We are also grateful for the Documso annotators for manually labeling the dataset.

REFERENCES

[1] M. Rusinol, T. Benkhelfallah, and V. Poulain dAndecy, "Field extraction from administrative documents by incremental structural templates," in *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 1100–1104.

[2] J. X. Zeng, D. G. Bi, and X. Fu, "A matching method based on svd for image retrieval," in *2009 International Conference on Measuring Technology and Mechatronics Automation*, vol. 1. IEEE, 2009, pp. 396–398.

[3] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1091–1095, 2007.

[4] X. M. Niu and Y. H. Jiao, "An overview of perceptual hashing," *Acta Electronica Sinica*, vol. 36, no. 7, pp. 1405–1411, 2008.

[5] O. Chum, J. Philbin, A. Zisserman *et al.*, "Near duplicate image detection: min-hash and tf-idf weighting." in *BMVC*, vol. 810, 2008, pp. 812–815.

[6] F. Cesarini, M. Gori, S. Marinai, and G. Soda, "Informys: A flexible invoice-like form-reader system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 730–745, 1998.

[7] V. P. d'Andecy, E. Hartmann, and M. Rusinol, "Field extraction by hybrid incremental and a-priori structural templates," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 251–256.

[8] D. Schuster, K. Muthmann, D. Esser, A. Schill, M. Berger, C. Weidling, K. Aliyev, and A. Hofmeier, "Intellix–end-user trained information extraction for document archiving," in *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 101–105.

[9] Rossum, "Rossum: Data extraction with artificial intelligence," Accessed on: Aug 27, 2019. [Online]. Available: https://rossum.ai

[10] M. Holeček, A. Hoskovec, P. Baudiš, and P. Klinger, "Line-items and table understanding in structured documents," *arXiv preprint arXiv:1904.12577*, 2019.

[11] X. Holt and A. Chisholm, "Extracting structured data from invoices," in *Proceedings of the Australasian Language Technology Association Workshop 2018*, 2018, pp. 53–59.

[12] R. B. Palm, O. Winther, and F. Laws, "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. IEEE, 2017, pp. 406–413.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
[15] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[16] Google, "Google Cloud Vision OCR," Accessed on: Aug 27, 2019. [Online]. Available: https://cloud.google.com/vision/docs/ocr
[17] Microsoft. Microsoft Azure Computer Vision. Accessed on: Aug 27, 2019. [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision
[18] Amazon Web Services. Amazon Textract. Accessed on: Aug 27, 2019. [Online]. Available: https://aws.amazon.com/textract
[19] A. Cohen, "FuzzyWuzzy: Fuzzy string matching in python," Accessed on: Aug 27, 2019, 2011. [Online]. Available: https://pypi.org/project/fuzzywuzzy
[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.