# MONSTERMASH: Multidirectional, Overlapping, Nested, Spiral Text Extraction for Recognition Models of Arabic-Script Handwriting

Danlu Chen[1][0000000285828151], Jacob Murel[2], Taimoor Shahid[3], Xiang Zhang[1], Jonathan Parkes Allen[3], Taylor Berg-Kirkpatrick[1], and David A. Smith[2]

[1] University of California, San Diego, CA, U.S.A.
[2] Northeastern University, Boston, MA, U.S.A.
[3] University of Maryland, College Park, MD, U.S.A.
dac013@ucsd.edu

**Abstract.** Most current models for handwritten text recognition transcribe individual lines and thus depend on accurate line extraction from page images. This line extraction task is particularly challenging for Arabic-script manuscripts, which exhibit a high proportion of curved lines, word baselines that vary within the line, and varying line orientation on the page. We present a new corpus for studying Arabic-script line extraction in the presence of these phenomena and evaluate different model architectures using several pixel-level, object-level, and extrinsic recognition metrics. Training all models on the same data, we find that the CNN-based Kraken model slightly outperforms the transformer-based TESTR model on recognition character accuracy and some object-level metrics, even though it lags behind on pixel-level metrics.

**Keywords:** Islamicate manuscripts · Text line extraction · HTR

## 1 Introduction

Methods for handwritten text recognition (HTR) have an array of applications for advancing research on historical manuscripts—from enabling full-text search of library collections, to conducting large-scale linguistic analyses of digitized documents. HTR has made significant progress since adopting line-level transcription approaches such as connectionist temporal classification (CTC). Just as earlier word-spotting approaches required segmenting a page image into word patches, line-level models require that the lines of text be extracted from a page to produce both training and test data. Although whole-page encoder-decoder models have been proposed, their successful deployment still requires, at the current state of the art, a set of phased training steps involving individual lines.

While there are many annotated datasets and shared-task evaluations to choose from for the task of transcribing images of single lines, annotated datasets and published evaluations of line extraction are less common. In any case, line extraction for handwritten text in Latin, Hebrew, Chinese, and other scripts

can achieve a high degree of accuracy. The effectiveness of line extraction for Arabic-script manuscripts—in Arabic, Persian, Ottoman Turkish, Urdu, and other languages—seems anecdotally to be lower and less well explored.

This paper, therefore, seeks to fill this gap and provide a systematic analysis of the effectiveness of different line-extraction approaches for Arabic-script manuscripts. In the literature, these models have been evaluated on scripts such as Latin and Hebrew with various levels of success. Our contributions are both *observational*, resulting in a compilation of existing and new annotated data, and *experimental*, evaluating the effectiveness of pixel-labeling and object-detection approaches and convolutional and transformer neural architectures.

To perform our evaluations, we compile existing annotated data for Arabic-script manuscripts and also produce a newly annotated test set. Unlike some existing annotated datasets, which may omit marginalia and confine themselves to main-text horizontal lines, our new test set marks all lines of written text on each page and balances samples from simpler and more complex page layouts.

We evaluate three line-extraction approaches, training and testing them on the same data. First, `kraken` [10] labels pixels on a page as belonging to a baseline and then, in a postprocessing step, heuristically infers a polygon that bounds the characters on that baseline. Second, `doc-ufcn` [5] labels pixels as belonging to a line region. Third, `TESTR` directly infers a polygon bounding a line region.

We find that the pixel-based `kraken` performs best according to the object-level AP@0.75 metric, while the detection-based method `TESTR` performs best according to pixel-level metrics. While `TESTR` is slower than our other models, it is also the most accurate on the most difficult line extraction examples: non-linear text with extreme curvature. In terms of downstream recognition accuracy, `TESTR` markedly outperforms `doc-ufcn`, but is marginally worse than `kraken`. Our results show that each system has its own strengths and weaknesses. We hope our provided datasets encourage further work on Arabic-script HTR for complex document layouts.

## 2 Related Work

### 2.1 Text Line Extraction

As mentioned, a critical initial step in most HTR pipelines is text line extraction. In the literature, there are three major approaches to line extraction: baseline-based [10], region-based [6], and detection-based [23] methods.

Baseline-based methods identify which pixels in the input image correspond to a baseline of a text line in the image. After the baselines have been identified, a post-processing step infers the surrounding polygon for each text line. Region-based methods also make pixel-level predictions, but instead of identifying baselines, they identify all pixels within a text line region. Detection-based methods, on the other hand, instead directly predict the vertices of polygons surrounding each text line.

The former two methods show suitable performance for horizontal lines, which has been the primary focus of past research [7, 21, 1]. Natural scene text

extraction has similarly focused on linear text [18, 12]. Non-linear lines, however, pose a significant challenge for these systems.

Previous approaches to detecting non-linear text lines have involved character-level bounding boxes [3] and semantic segmentation tools [15, 13]. While these approaches show promise, they require computationally expensive pre- and post-processing steps. By contrast, transformer-based text-detection promises an effective and less expensive approach for non-linear text line extraction [23, 9, 20]

### 2.2 Arabic Script HTR

As with much language-focused research, HTR experiments have traditionally focused on Latinate texts—e.g. English, Spanish, French, etc. In the past few years, however, researchers have turned to non-Latinate documents. Arabic-script HTR poses several unique challenges due to its connecting script, use of diacritical marks, and changing character forms [14, 11].

Existing Arabic-script HTR datasets include IFN-ENIT databse[4], AHDB [2], and KHATT [16]. Each of these, however, consists of images of text regions demarcated at the word or paragraph level sourced from modern writers for HTR research. As such, text lines are straight and undisturbed by peripheral text regions occupying image space.

Other studies that use datasets of historical Arabic-script documents do show promising work for Arabic-script text line extraction [21, 1]. Unfortunately, these studies similarly focus on documents with exclusively horizontal lines. Moreover, neither publicly release their respective datasets for use, making it difficult to replicate and build upon their work.

We know of only two other study that exclusively addresses non-linear text line extraction for handwritten Arabic-script. [22] proposes partial contouring and projection to recognize curved text lines as horizontal segments and concatenate these segments for transcription. This approach evidences useful results. Given its dated-ness, however, it does not take advantage of the significant developments in test line extraction from the past decades. [4] also collected a curved text line dataset but only pixel-wise labels are available.

## 3 Dataset Description

### 3.1 Dataset Overview

The different subsets of ITI-bench are summarized in Table 1. There are two subsets for training and the other two for testing. We estimated the quality of the data by sampling 20 pages from each dataset and manually counted the number of missing lines or incorrectly labelled lines. We calculated the missing ratio for main body text (main) and marginalia separately, which is reported in the last two column in Table 1. 7.5% of the main body text is missing in the `arabic_ms` set.

[4] `http://www.ifnenit.com/`

| | | pages | | | | | drop rate | |
|---|---|---|---|---|---|---|---|---|
| **subset** | **usage** | **bsln.** | **poly.** | **HT** | **trans.** | **lines** | **body** | **margin** |
| **aocp_print** | train | 2,970 | 2,718 | No | Yes | 67,750 | 2.7% | 17.3% |
| **arabic_ms** | train | 873 | 873 | Yes | Yes | 16,635 | 7.5% | 15.7% |
| **aocp_ms_eval** | test | 113 | 112 | Yes | Yes | 2,025 | 0.0% | 22.4% |
| **iti-complex** | test | 52 | 52 | Yes | No | 2,536 | 0.0% | 0.8% |

**Table 1.** Overview of dataset statistics. **bsln.** standards for number of pages have valid baseline annotation and **poly.** for number of pages that have valid polygon annotation.

1. `aocp_print`, annotated printed documents, total about 2,970 pages. Only 2,718 of them can be converted into polygons automatically.
2. `arabic_ms`, 873 annotated pages.
3. `aocp_ms_eval`, 113 annotated pages with transcription, `https://github.com/OpenITI/aocp_ms_eval`
4. `iti-complex`, a complex layout selection of Islamicate manuscripts.

### 3.2 Annotation protocol

The `iti-complex` subset is a newly annotated set of line extractions that we provide. Our goal was to provide a more challenging benchmark for line extraction in this domain that more accurately represents the diversity of page layouts and line orientations. Thus, `iti-complex` consists of a set of 52 single page images selected from a wide range of manuscripts held within a diversity of digital repositories large and small. We aimed to represent multiple vectors of that diversity as much as possible within the constraints of the data set, placing an emphasis on manuscript pages with high degrees of layout complexity and line and word-form variability (that is, complex ligatures, full vocalization, tall ascending and descending letter forms, and so forth). We also captured a high degree of chronological, regional, and linguistic diversity, sampling not only Arabic and Persian but also Ottoman Turkish, Urdu, and Uyghur texts, along with multi-script texts including Coptic, Sanskrit, and Latin scripts. Generic diversity was somewhat constrained by our privileging high complexity layouts, although there too we included a wide range of texts. Finally, we also sought to represent the diversity of digitized exemplars, which range from high quality scans typical of recent digitization efforts to much lower quality, lower resolution and often grey scale or even black-and-white scans that are more typical of early digitization efforts or represent the digitization of microfilm.

Each page in this set is annotated with a polygon of up to 16 vertices using Labelme [19] by our experts. After selecting a page for use in the dataset, we saved a JPEG or PNG file, added the metadata relevant to the manuscript in question along with brief descriptions of layout features, and opened the image in Labelme. While we did practice some export and editing of previously annotated images (generated using the transcription platform eScriptorium, and

**Fig. 1.** Example pages from `iti-complex` (image id 3, 6, 20, 23, 37, 49). There are not only vertical or diagonal aligned text lines, but also spiral, curved text lines.

then geometrically simplified), most of the material in our data set was annotated using previously un-annotated images taken directly from publicly open online repositories. We then drew the polygons within the application, maintaining a maximum of sixteen points, and including all letter-forms and orthographic devices as well as 'punctuation' marks. We treated words within a continuous line as units for polygon generation, which entailed in some cases breaking up continuous semantic units which however were not spatially continuous. In general this approach allowed us to capture the entirety of the letter-forms while minimizing neutral space and overlap with lines above and below (and on occasion intersecting in some way from the left or right). On the other hand, we treated lines in two columns that were a semantic unit as one continuous line. It is common in the Islamicate poetic tradition, which frequently employs two-line poetry units, to inscribe the two lines in separate columns. However, these are not true columns and are ignored while reading the text as the reading order involves reading the first line from the right column and jumping to the next column (left), and then jumping back to the first column for the next line. As such, for these cases, we treated the two lines in two columns as one. On occasion

we were forced to make educated guesses about the relationship of a given dot to two lines, a not uncommon occurrence in Arabic script, but such instances were rare. Other edge cases included lines in which poor ink quality or damage to the manuscript over time made the precise decipherment of a line and its letter-forms difficult if not impossible. Finally, we had to make determinations about the occasional use of superscript letters used for in-text annotation and other purposes, choosing to include them as part of the line which they modified.

Figure 1 shows several examples from the ITI-complex subset.

## 4 Methods

There are three major kinds of line extraction methods: *baseline-based*, *region-based*, and *detection-based*. Both *baseline-based* and *region-based* methods produce pixel-wise predictions: *baseline-based* approaches identify pixels that represent the baseline of each text line, while *region-based* methods identify pixels that makeup the entire text line region. (In the computer vision community, models that predict dense pixel-wise outputs are also called segmentation-based methods.) In contrast, *detection-based* methods predict the vertices of polygons that contain each individual text line. In order to compare these paradigms, we train and test three representative line extraction models: kraken (baseline-based) [10], doc-ufcn (region-based) [6] and TESTR (detection-based)[23].

| | type | output | overlap | post-process | backbone | train iter. |
|---|---|---|---|---|---|---|
| **kraken** | baseline | binarized pixels | No | Yes | convnet | <50,000 |
| **doc-ufcn** | region | binarized pixels | No | Yes | convnet | 10,000 |
| **TESTR** | detection | vertices of polygon | Yes | No | transformer | 200,000 |

**Table 2.** A summary of the three kinds of line extraction methods. The **overlap** column indicates whether the method can represent overlapping line regions, while the **post-process** column indicates whether the backbone model's output requires post-processing in order to specify lines. Finally, the**train iter.** column shows the number of training iterations needed for each model.

### 4.1 Kraken and DOC-UFCN

Kraken is an open-source OCR system that specializes in historical and non-Latin scripts. Both Kraken's line detection module and DOC-UFCN are U-NET-based models which consist of sequences of convolutional layers followed by transposed convolutional layers. A U-Net-like model generates 2D pixel-wise predictions that maintain the same dimensions as the input. Each pixel receives a numerical prediction ranging from 0 to 1. Kraken specifically predicts baselines, whereas DOC-UFCN classifies all pixels within the text line polygon as positive. For post-processing, Kraken employs heuristic methods to expand the baseline into a bounding box, while DOC-UFCN identifies connected components in the 2D prediction heatmap to extract polygons.

### 4.2 TESTR

TESTR is a transformer-based text detection model and predicts vertices of polygons directly. As the predictions does not restricted to be non-overlapping, or sometimes they can be identical, therefore we need to remove the redundant polygons. We perform non-maximum suppression (NMS) to reduce overlapping polygons and keep only the polygons with highest scores.

## 5 Experiments

For Kraken, we inference on the model checkpoints released by [17]. For the rest two models, we trained from scratch using similar training procedure.

### 5.1 Pre-processing

*DOC-UFCN* requires shrunken polygon to achieve the best performance. We set the shrink ratio to 0.1. We resize the image so that the width or height is no longer than 768 px for training data.

*TESTR* requires polygons with no more than 16 vertices. There are usually more than 40 vertices for each polygon for all subsets expect the **iti-complex** subset. We use the Douglas-Peucker algorithm[8] to simply the polygon and then compute the convex hull of the polygon.

### 5.2 Training

For DOC-UFCN and TESTR training, we followed Kraken's training recipe for fair comparison. We first train on `AOCP_print` (phase I) and then fine-tuning on `arabic_ms` (phase II).

**DOC-UFCN** We trained from scratch with a learning rate $5^{-5}$ and batch size 32. We trained for $10,000$ iterations.

**TESTR** In Phase I, we trained from scratch with a learning rate $5^{-4}$ and batch size 8. We trained for $200,000$ iterations. In Phase II, we trained on the last checkpoints from Phase I with a learning rate $5^{-5}$ and batch size 8. We trained for $200,000$ iterations.

### 5.3 Inference

**Kraken** If any side of the image is greater than 3000px, we need to resize it first to feed to Kraken. We simply using the default setting of Kraken for inference.

**DOC-UFCN** We use the default setting of the model for inference.

**TESTR** During influence, we set the confidence threshold to 0.05, i.e. we ignore predictions whose confidence scores are lower than 0.05. We perform non-maximum suppression (NMS) to reduce overlapping polygons. The IoU threshold for running NMS is 0.3.

### 5.4 Evaluation metrics

Following [7], we use both *pixel-level* and *object-level* metrics for intrinsic evaluation of extracted lines, while we use *downstream OCR performance* as an extrinsic evaluation. In contrast with [7], however, we use macro-averaging across lines in all pixel-based metrics (i.e. those that compare areas of regions) to avoid issues arising from double counting overlapping line regions.[5]

**Pixel-level metrics.** These metrics are also used widely in general semantic segmentation tasks in the computer vision community. They evaluate the prediction performance aggregating across individual pixels in predicted and ground truth regions. However, these metrics sometimes fail to reflect whether a complete line region has been detected accurately.

1. **(macro) IoU.** For each page, we first compute the union of all the predicted line regions, `predict_union`, and similarly, compute the union of all ground truth line regions, `label_union`. Finally, the intersection over union (IoU) metric is given by calculating the ratio of the areas of the intersection and union of `predict_union` and `label_union`.
2. **(macro) recall/precision/$\mathbf{F}_1$.** Similarly, recall is given by computing the ratio of the area of the intersection of `predict_union` and `label_union` with the area of `label_union`, while precision is calculated in a similar fashion but using the area of `predict_union` as the denominator. $\mathrm{F}_1$ is the geometric mean of precision and recall.

**Object-level metrics.** These metrics are commonly used to evaluate object detection tasks in computer vision. We first greedily map predicted line regions to ground truth regions to maximize IoU, and then set an IoU threshold to determine which predicted regions can be considered correct. After determining the correct and incorrect predicted line regions based on the chosen threshold, we compute object-level precision, recall, and $\mathrm{F}_1$. By varying the IoU threshold, we can compute metrics with variable tolerance in determining 'correctness'. Further, by varying the underlying threshold of model score that determines the confidence level at which the model makes a prediction, we can measure the tradeoff between precision and recall.

[5] Alternatively, it would also be possible to align predict and groundtruth regions first, and then calculate micro-averaged pixel-based metrics. However, this cause the pixel-based metrics somewhat redundant with the separate object-based metrics we calculate.

1. **AP (Average Precision)**. This commonly used metric in the objective detection literature is derived by plotting the object-level precision-recall curve while sweeping the model confidence threshold across its full range and then computing the area under this curve. AP@.5 refers to the area under the precision-recall curve when the intersection over union (IoU) threshold is set to 0.5.

**OCR performance.** Finally, as an extrinsic metric of line extraction performance, we pass the extractions of the pedicted line regions to a downstream OCR engine and measure OCR performance against a groundtuth transcription. Specifically, we calculate the character accuracy rate (CAR) for each line extraction model using the recognition model from [17] as an OCR engine.

| | pixel-level | | | | object-level | | OCR |
|---|---|---|---|---|---|---|---|
| | **IOU** | **P** | **R** | **$F_1$** | **AP@.5** | **AP@.75** | **CAR** |
| **kraken** | 83.04 | 98.39 | 80.95 | 87.77 | 68.27 | **56.49** | **65.08** |
| **doc-ufcn** | 62.93 | 94.29 | 65.78 | 76.78 | 68.55 | 13.23 | 47.42 |
| **TESTR** | 91.21 | 87.00 | 98.95 | 92.60 | **76.74** | 32.09 | 63.83 |
| **manual** | | | | | | | 77.94 |

**Table 3.** Result on `aocp_ms_eval`. **Manual** reports the CAR when feeding the recognition system with manually labelled gold bounding boxes as a reference.

### 5.5 Results

The results on three subsets of the ITI dataset. Table 3 shows performance on `aocp_ms_eval`. As summarized in Table 3, about 25% of the annotation for marginalia is missing and therefore, the performance of TESTR–the method can predict fairly good on marginalia is penalized, as shown in Fig 2. Note that TESTR predict the polygon directly, therefore, the IOU is calculated by the the IOU of the union of predicted polygons and the union of labels.

We run the recognition model from [17] on the exacted lines from different models and report the character accuracy rate (CAR). We take the average of CAR from each page. Note that some of the labelled data does not have any manual transcription, and we just skip it.

| | pixel-level | | | | object-level | |
|---|---|---|---|---|---|---|
| | **IOU** | **Precision** | **Recall** | **$F_1$** | **AP@.5** | **AP@.75** |
| **kraken-ft** | 45.22 | 95.37 | 45.48 | 57.16 | 47.16 | **24.39** |
| **doc-ufcn-ft** | 51.01 | 91.75 | 54.31 | 66.67 | 36.03 | 1.03 |
| **TESTR-ft** | 83.60 | 91.99 | 88.62 | 87.89 | **51.66** | 17.71 |

**Table 4.** Result on `iti-complex`.

## 6 Analysis and Discussion

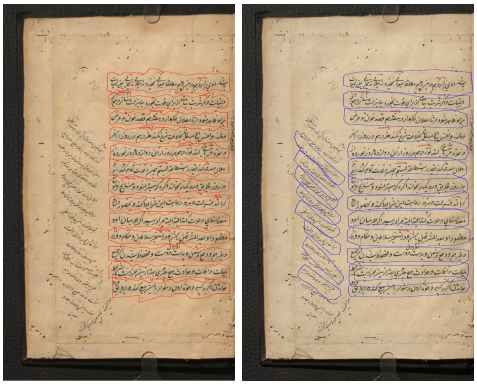

**Fig. 2.** Annotations sometimes miss the marginalia, which increases the false positive rate in models such as TESTR. **Left**: the annotation. **Right**: TESTR's prediction.

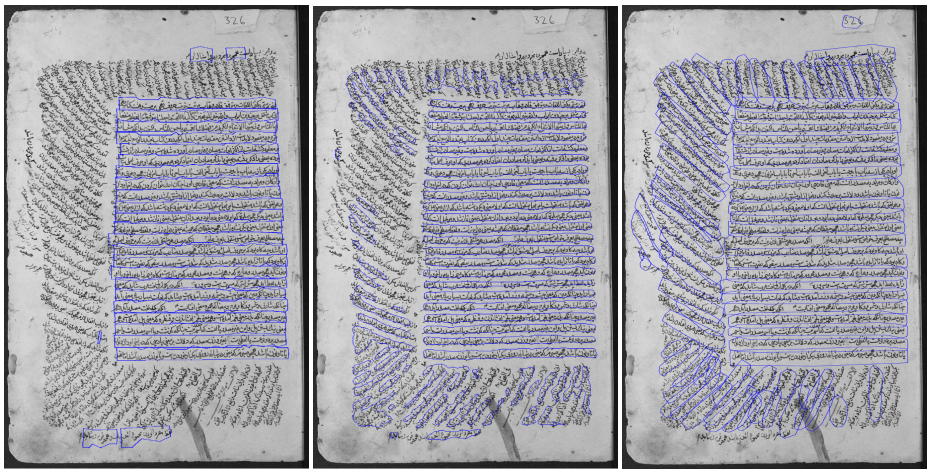

**Fig. 4.** Prediction of `aocp_ms_eval` (image id 46), From left t o right: Kraken, DOC-UFCN, TESTR. All three models predict most of the main-text body and Kraken works best on main-text body. TESTR recalls most of the text in marginalia.

We analyse the experimental results and summarize them into the following key aspects.

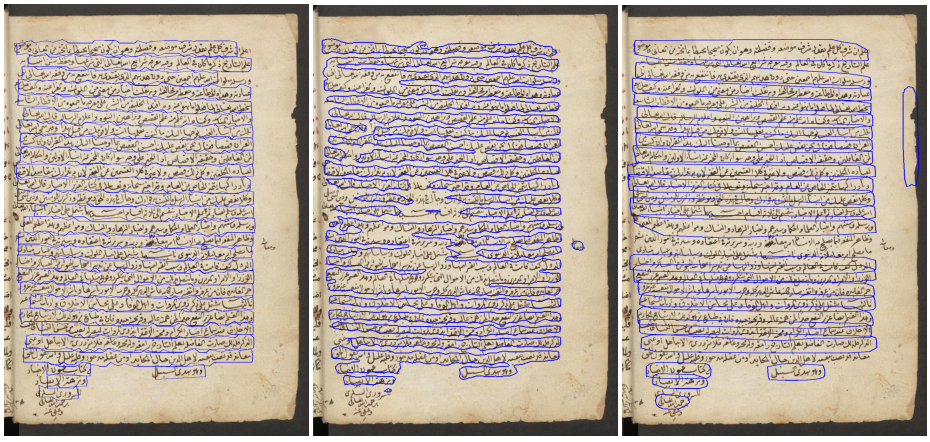

**Fig. 3.** Prediction of `aocp_ms_eval` (image id 47). From left to right: Kraken, DOC-UFCN, TESTR. DOC-UFCN is able to capture text lines in main text body, but the adjacent lines can not be differentiated.

1. Kraken works best on regular, dense, main-body text. It sometimes fail to capture the full line. It mostly fails to handle outliers. We also observe that Kraken achieve best number for the AP@.75 metric on both test sets. We argue it may be because Kraken use a comparably heavy post-processing algorithm to obtain the final bounding boxes.
2. DOC-UFCN performs very bad on iti-complex mainly because it cannot predict clean single lines when lines are irregular.
3. Overall, TESTR achieved the best performance in term of recall and AP@0.5. This indicates that TESTR is more robust to out-of-distribution new examples. As shown in Fig 3 and Fig 4, TESTR retrieved most of the marginalia text. However, we notice that the TESTR is expected to have uniform performance on similar text blocks on same page while TESTR usually drop one line. Another issue we found on TESTR is that the confidence of predicted polygons is low within $[0.1, 0.3]$ (shown in Fig. 5). Interestingly, the average confidence on `iti-complex` is significantly larger than that on `arabic_ms_eval`.
4. The CAR (Character Accuracy Rate) for DOC-UFCN is notably low. As observed, DOC-UFCN struggles to differentiate between closely spaced lines, often predicting adjacent lines as a single polygon. To address this issue, additional post-processing efforts could be implemented to enhance performance without necessitating modifications to the neural network model.
5. TESTR is approximately 8 times slower in training compared to the other two models, primarily due to its use of a Transformer-based backbone. Conversely, during inference, Kraken exhibits the slowest performance owing to its extensive post-processing requirements.

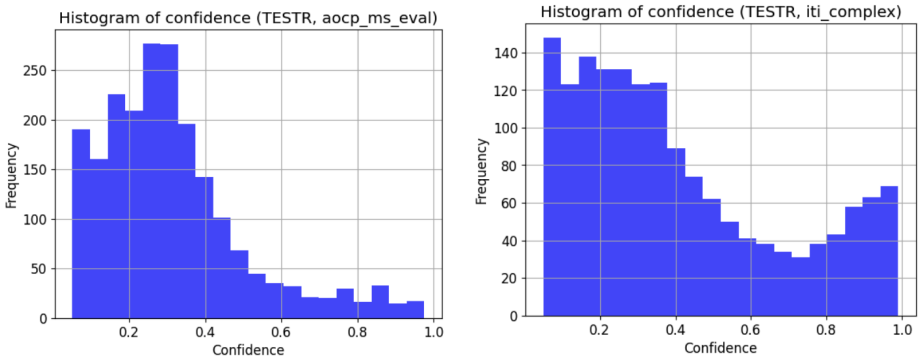


**Fig. 5.** Histograms of confidence scores of polygons predicted by TESTR on two test sets. The majority of the confidence scores are lower than 0.5.

## 7 Case study: How Noisy data Affects Performance

As we observed, the annotations in the training dataset, specifically in `arabic_ms_data`, are incomplete with approximately 7.5% of the main body text lines missing, and the missing rate for marginalia are doubled to 15.7%. Additionally, we note that the confidence scores for the predicted polygons from TESTR are lower than expected (shown in Fig. 5). These two observations prompt further investigation into how such noisy data influences model performance.

We also tested checkpoints before the model loss.

We randomly drop 20% and 50% of the polygons in `arabic_ms_data` before beginning the fine-tuning phase (phase II) and then evaluate the impact on test performance using both `aocp_ms_eval` and `iti-complex` datasets. As anticipated, there was a significant deterioration in performance with a 20% reduction in annotations. Surprisingly, however, performance did not decrease further when we removed an additional 30% of the annotations, as shown in Table 5. The `AP@.5` metrics barely changed when we reduced the number of annotated lines, when we manually look at some of the prediction, .

| | `arabic_ms_data` | | | `iti-complex` | | |
|---|---|---|---|---|---|---|
| **drop rate** | **0%** | **20%** | **50%** | **0%** | **20%** | **50%** |
| **AP@.50** | **76.74** | 73.11 | 70.98 | **51.66** | 43.32 | 43.19 |
| **AP@.75** | **32.09** | 18.24 | 18.00 | **17.71** | 13.76 | 13.58 |
| **avg. confidence** | **31.36** | 26.45 | 22.81 | **40.67** | 30.48 | 27.75 |

**Table 5.** Ablation study on noisy data with a drop rate of 0 %, 20%, 50%.

## 8 Conclusion

We empirically evaluate three major methods for text line extraction on Islamicate documents: baseline (`kraken`), region (`doc-ufcn`), and detection-based (`TESTR`). Scores for all three methods decrease drastically when tested exclusively on non-linear lines (i.e. the `iti-complex` dataset).

Evaluated at the object level, we find that a pixel-based model such as `kraken` works best on AP@0.75, while a detection-based method like `TESTR` performs better on AP@0.5. Indeed, `TESTR` achieves the highest performance for non-linear text line extraction in terms of recall and AP@0.5. Both `TESTR` and `kraken` significantly outperform `doc-ufcn` with respect to OCR CAR. Evaluated at the pixel level for F1, `TESTR` notably outperforms both `kraken` and `doc-ufcn`.

Our comparative evaluation reveals that each system has its unique strengths and weaknesses. Nevertheless, we believe these results also evidence the potential for transformer-based approaches to non-linear text line extraction.

## References

1. Al-Barhamtoshy, H.M., Jambi, K.M., Abdou, S.M., Rashwan, M.A.: Arabic documents information retrieval for printed, handwritten, and calligraphy image. IEEE Access **9**, 51242–51257 (2021)
2. Al-Ma'adeed, S., Elliman, D., Higgins, C.: A data base for arabic handwritten text recognition research. In: Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. pp. 485–489 (2002)
3. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. p. 9365–9374 (2019)
4. Barakat, B.K., Cohen, R., El-Sana, J.: Vml-moc: Segmenting a multiply oriented and curved handwritten text line dataset. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW). vol. 6, pp. 13–18. IEEE (2019)
5. Boillet, M., Kermorvant, C., Paquet, T.: Multiple document datasets pre-training improves text line detection with deep neural networks. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 2134–2141. IEEE Computer Society, Los Alamitos, CA, USA (jan 2021). https://doi.org/10.1109/ICPR48806.2021.9412447, `https://doi.ieeecomputersociety.org/10.1109/ICPR48806.2021.9412447`
6. Boillet, M., Kermorvant, C., Paquet, T.: Multiple Document Datasets Pre-training Improves Text Line Detection With Deep Neural Networks. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 2134–2141 (Jan 2021). https://doi.org/10.1109/ICPR48806.2021.9412447
7. Boillet, M., Kermorvant, C., Paquet, T.: Robust text line detection in historical documents: learning and evaluation methods. In: International Journal on Document Analysis and Recognition (IJDAR). pp. 1433–2825 (Mar 2022). https://doi.org/10.1007/s10032-022-00395-7, `https://doi.org/10.1007/s10032-022-00395-7`

8. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: the international journal for geographic information and geovisualization **10**(2), 112–122 (1973)
9. Huang, M., Zhang, J., Peng, D., Lu, H., Huang, C., Liu, Y., Bai, X., Jin, L.: Estextspotter: Towards better scene text spotting with explicit synergy in transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 19495–19505 (October 2023)
10. Kiessling, B.: The Kraken OCR system (Apr 2022), `https://kraken.re`
11. Lamsaf, A., Aitkerroum, M., Boulaknadel, S., Fakhri, Y.: Text line and word extraction of arabic handwritten documents. In: Ben Ahmed, M., Boudhir, A.A., Younes, A. (eds.) Innovations in Smart Cities Applications. pp. 492–503 (2019)
12. Liao, M., Shi, B., Bai, X., Wang, X., Liu, W.: Textboxes: A fast text detector with a single deep neural network. In: Proceedings of the thirty-first AAAI conference on artificial intelligence. pp. 4161–4167 (2017)
13. Long, S., Ruan, J., Zhang, W., He, X., Wu, W., Yao, C.: Textsnake: A flexible representation for detecting text of arbitrary shapes. In: European Conference on Computer Vision (ECCV). p. 19–35 (2018)
14. Lorigo, L., Govindaraju, V.: Segmentation and pre-recognition of arabic handwriting. In: Eighth International Conference on Document Analysis and Recognition (ICDAR). pp. 605–609 (2005)
15. Lyu, P., Liao, M., Yao, C., Wu, W., Bai, X.: Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
16. Mahmoud, S.A., Ahmad, I., Alshayeb, M., Al-Khatib, W.G., Parvez, M.T., Fink, G.A., Märgner, V., Abed, H.E.: Khatt: Arabic offline handwritten text database. In: 2012 International Conference on Frontiers in Handwriting Recognition. pp. 449–454 (2012)
17. Smith, D.A., Murel, J., Allen, J.P., Miller, M.T.: Automatic collation for diversifying corpora: Commonly copied texts as distant supervision for handwritten text recognition. In: Computational Humanities Research Conference (CHR) (2023)
18. Tian, Z., Huang, W., He, T., He, P., Qiao, Y.: Detecting text in natural image with connectionist text proposal network. In: Proceedings of the European conference on computer vision. p. 56–72 (2016)
19. Wada, K.: Labelme: Image Polygonal Annotation with Python. https://doi.org/10.5281/zenodo.5711226, `https://github.com/wkentaro/labelme`
20. Ye, M., Zhang, J., Zhao, S., Liu, J., Du, B., Tao, D.: Dptext-detr: Towards better scene text detection with dynamic points in transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 3241–3249 (2023)
21. Zahour, A., Likforman-Sulem, L., Boussalaa, W., Taconet, B.: Text line segmentation of historical arabic documents. In: Ninth International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 138–142 (2007)
22. Zahour, A., Taconet, B., Mercy, P., Ramdane, S.: Arabic hand-written text-line extraction. In: Proceedings of Sixth International Conference on Document Analysis and Recognition. pp. 281–285 (2001)
23. Zhang, X., Su, Y., Tripathi, S., Tu, Z.: Text spotting transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9519–9528 (June 2022)