Improving Information Extraction from Visually Rich Documents using Visual Span Representations

Ritesh Sarkhel
The Ohio State Universtiy
Columbus, Ohio
sarkhel.5@osu.edu

Arnab Nandi
The Ohio State University
Columbus, Ohio
nandi.9@osu.edu

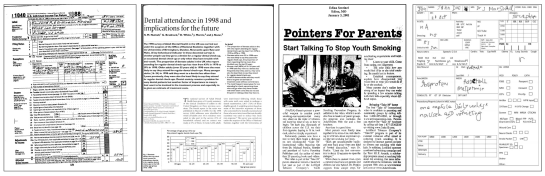
ABSTRACT

Along with textual content, visual features play an essential role in the semantics of visually rich documents. Information extraction (IE) tasks perform poorly on these documents if these visual cues are not taken into account. In this paper, we present `Artemis` – a visually aware, machine-learning-based IE method for heterogeneous visually rich documents. `Artemis` represents a visual span in a document by jointly encoding its visual and textual context for IE tasks. Our main contribution is two-fold. First, we develop a deep-learning model that identifies the local context boundary of a visual span with minimal human-labeling. Second, we describe a deep neural network that encodes the multimodal context of a visual span into a fixed-length vector by taking its textual and layout-specific features into account. It identifies the visual span(s) containing a named entity by leveraging this learned representation followed by an inference task. We evaluate `Artemis` on four heterogeneous datasets from different domains over a suite of information extraction tasks. Results show that it outperforms state-of-the-art text-based methods by up to 17 points in F1-score.

PVLDB Reference Format:
Ritesh Sarkhel and Arnab Nandi. Improving Information Extraction from Visually Rich Documents using Visual Span Representations. PVLDB, 14(5): 822 - 834, 2021.
doi:10.14778/3446095.3446104

1 INTRODUCTION

A significant number of documents we encounter every day are visually rich in nature. Along with linguistic cues, they employ several explicit (e.g., relative positioning, font-size, font-color) and implicit (e.g., negative distance, whitespace balance) visual features to augment or highlight the information they disseminate. Whether ordering from a *restaurant-menu*, comparing properties in real-estate *flyers*, or looking up events in a *poster*, the information contained in these documents is often ad-hoc i.e., they cannot be retrieved from an indexed database quite easily. Text-based methods fail to address the challenges of information extraction (IE) in these scenarios, as they do not consider the contribution of visual features on the semantics of these documents, thus leaving the user with little choice than mentally parsing the entire document to extract relevant information. What makes the task of automated extraction more challenging is the heterogeneous nature of these documents. Real-world documents can be diverse in terms of layout, format, and content. These challenges accumulate to motivate the need for a generalized method for IE from visually rich documents. Before we present our method, to explain our contributions better, we demonstrate the limitations of a text-only method for IE from visually rich documents in an example below.



(a) Tax-form 1040 (b) Biomedical Journal (c) News article (d) Brain document

Figure 1: Samples of visually rich documents. Visual features play an important role in augmenting their semantics

Example 1.1: Alice, an analyst studying tobacco addiction, wants to perform a longitudinal study on the coverage of adverse effects of tobacco usage in print-media. She has collected some articles from national newspapers and magazines on this topic published in the last three decades. The documents in Alice's collection are visually rich and heterogeneous in nature i.e., they have diverse layouts, formats, and content. While most recently published articles are born-digital and available in HTML format, the older articles are only available as high-resolution scanned copies. Alice wants to extract three named entities from each document in her collection: N = {'Name of the Tobacco product', 'Manufacturer's Name', 'Listed side effects'}. For each named entity $n_i \in N$, $i = \{1, 2, 3\}$, she wants to identify where in the document the named entity appears, transcribe it, and store it in a CSV file. A text-based IE method, in such scenarios, typically starts with cleaning the document first. This includes perspective warping, skew correction, and blurring. The document is then transcribed, its text is normalized, and stopwords are removed. Finally, named entities are extracted by identifying spans of text that contain specific syntactic or semantic patterns representing a named entity $n_i \in N$.

Challenges: Although reasonable for plain-text documents, following a similar approach Alice's collection can be challenging. *First*, if the document layout is not known beforehand, text-based extractors do not always lead to a reliable extraction performance [51]. Errors introduced during optical character recognition and serialization of the transcribed text lead to incorrect semantic parsing, which affects the downstream IE task. Furthermore, the contribution of visual features (see Fig. 1) on the semantics such as relative positioning and font-size similarity are often not captured during

Proceedings of the VLDB Endowment, Vol. 14, No. 5 ISSN 2150-8097.
doi:10.14778/3446095.3446104

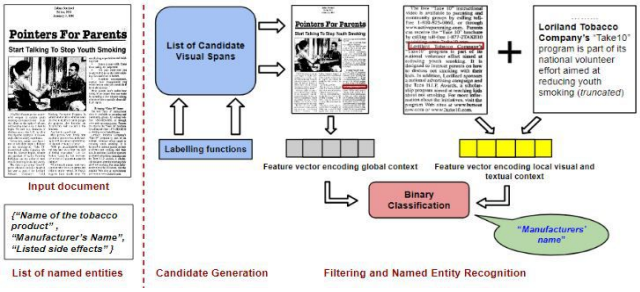


Figure 2: Overview of `Artemis`' extraction workflow. It takes a document D & a list of named entities N as input, and generates a list of candidate visual spans. Each candidate span (shown within rectangle) is represented using context vectors in a multimodal encoding space. The span containing a named entity is identified by an inference task

transcription. This motivates the need to encode a *visual span* (defined in Section 3) in the document by taking both its textual and visual modality into account. *Second*, a generalized method for information extraction should be robust [48] towards diverse layouts and formats (e.g., XML, HTML). It should also be flexible enough to be extended for various extraction tasks on different datasets.

Our hypotheses: Recent works have shown the importance of incorporating context information on the performance of text-based IE methods. From carefully designed syntactic/semantic features [29, 58, 60] to deep neural networks [25], state-of-the-art performances have been reported by encoding contextual signals from phrase-level, sentence-level, and paragraph-level abstractions of a text document. We postulate that incorporating similar contextual cues can boost extraction performance from visually rich documents too. It is well-known that the human cognitive system integrates context information from multiple modalities [43] to make sense of real-world data. We hypothesize that mimicking this behavior to encode multimodal context information can help address some of the challenges faced by text-based IE methods mentioned before. For example, a visual span appearing near the top of a news article is likely to be a date field. Therefore it is more likely to be transcribed as "11 Nov" than "ll Gov". We argue that representing a visual span by encoding such contextual cues can enhance the performance of IE methods on such documents. Unfortunately, while deep-learning models such as recurrent neural networks [1] are effective in modeling context information for text documents, they fall short if contextual cues span across both textual and visual modalities [23]. Following Doan et al. [9], we define the IE task addressed in this paper as a key-value retrieval task on the input document. Each key represents a named entity to be extracted, and its value denotes the smallest visual span that contains a true mention (defined in Section 3) of that named entity.

Contributions and Overview: We present `Artemis` – a generalized method for information extraction from heterogeneous, visually rich documents in this paper. It takes a document D, and a list of named entity types N as input and returns the smallest visual span in the document that contains a named entity $n_i \in N, \forall i$. `Artemis` works in two phases. First, it identifies a set of candidate visual spans in D that are likely to contain the named entities $n_i \in N, \forall i$. Then, it identifies the smallest visual spans within these candidates that contain the named entities in the second phase of its workflow. We represent each visual span using two fixed-length context vectors in our workflow. To encode its local context information, we extend a bidirectional long short-term memory (LSTM) network (detailed in Section 4.3) - the de-facto deep-learning standard in natural language processing [33]. It encodes both visual and textual properties of a visual span from its *local context boundary* (defined in Section 3). To identify the local context boundary of a visual span , we develop a machine-learning model [12] based on an adversarial neural network (detailed in Section 4.2) that incorporates domain-specific knowledge to identify the visual cues that act as local concept boundaries. An overview of our extraction workflow is presented in Fig. 2. `Artemis` builds upon existing works that either do not incorporate domain-specific contextual cues [51] or relies on format/layout-specific [4, 28, 59] features to identify named entities in a visually rich document. Contrary to these efforts, we do not leverage any format-specific features or assume explicit knowledge about the document layout at any stage of our workflow. Through exhaustive experiments on four cross-domain datasets we show that: (a) existing deep-learning models tailored for text-based information extraction e.g., long short-term memory (LSTM) networks [1] struggle to capture the multimodality of visually rich documents, and (b) ingesting domain-specific information to learn visual span representation boosts end-to-end extraction performance. Our main contributions are as follows:

- We describe `Artemis`, a visually-aware IE method for heterogeneous visually rich documents
- We propose a contextualized representation for a visual span in the document. To identify the context boundary of a visual span while incorporating domain-specific knowledge, we develop a machine-learning model based on an adversarial neural network.
- Through exhaustive evaluation on four heterogeneous datasets for separate tasks, we show that IE using our contextualized span representation outperforms text-only baselines

We formalize the IE task addressed in this paper in Section 2, and describe our method in Section 4. We evaluate our method on four separate IE tasks and discuss our findings in Section 5.

2 PROBLEM FORMULATION & DEFINITION

Given a document D and a list of named entity types $N = \{n_1, ..., n_k\}$ as input, our objective is to identify the smallest visual spans in D that contains a named entity in N. We formulate this information extraction task as a two-phase process. In the first phase, we identify a set of candidate visual spans (V_i) in D that are likely to contain a named entity $n_i \in N, \forall i$. We search for the smallest visual span(s) within V_i that contains the named entity $n_i \in N$ during the second phase of our workflow. Therefore formally, we define our information extraction task as a composition of two sub-tasks.

1. First sub-task: For each named entity type $n_i \in N, \forall i$, identify a set of candidate visual spans V_i in D that may contain n_i.

2. Second sub-task: For every named entity type $n_i \in N, \forall i$, find the smallest visual span within each V_i that contains n_i.

For the first sub-task, we employ a number of weakly supervised [44, 45] functions (Section 4.2). We formulate the second sub-task as a binary classification problem (Section 4.3) in a multimodal encoding space. Before presenting our extraction workflow, we review some of the background concepts in the following section first.

3 BACKGROUND CONCEPTS & DEFINITIONS

A. Visual Span: A visual span represents a contiguous area in a visually rich document. Assuming a rectangular coordinate system, we can represent a visual span as the tuple $\{x, y, w, h\}$. Here, (x, y) represents the coordinates of the left-top corner of the smallest bounding-box that encloses the visual span, w represents the width of the bounding-box, and h represents its height. The *local context boundary* of a visual span is the smallest, semantically coherent area in the document that contains that visual span (see Fig. 4).

B. Named Entity Recognition: We describe named entity recognition tasks by adopting terminology from the natural language processing community [22]. Two types of objects play central roles in a named entity recognition task: (a) *entity* and (b) *entity mention*. An entity e represents a distinct real-world person, place, or object. Entities can be grouped into different entity types. A mention v is a visual span containing text that refers to an entity. A candidate visual span or candidate span represents a non-empty visual span in the document that may contain a potential instance of an entity. If a candidate visual span is classified as true, it is deemed to be a mention. The input to a named entity recognition task is a visually rich document D and a list of entity types $N = \{n_1, n_2..., n_k\}$. It outputs the smallest visual span(s) that contains entity mentions of a named entity $n_i \in N$ in D.

C. Adversarial Neural Network: Given an input distribution Φ and a target distribution ζ, the goal of an adversarial network is to learn a transformation function f such that $\Phi : f \rightarrow \zeta$. For most real-world applications, the true form of the input distribution and/or the target distribution is typically not known beforehand. In such cases, the transformation function f is approximated using a pair of neural networks trained on separate but complementary learning objectives. Given a finite set of data-points S_Φ sampled from the input distribution Φ and their respective representations S_ζ in the target distribution ζ, the learning objective of one of the networks, referred to as the *Generator network* to learn a function f_g that transforms S_Φ to S_ζ i.e., $S_\Phi : f_g \rightarrow S_\zeta$. The second network referred to as the *Discriminator network*, on the other hand, learns an inverse function f_d to transform the data-points S_ζ sampled from the target distribution ζ to their corresponding representations S_Φ in the input distribution Φ i.e., $S_\zeta : f_d \rightarrow S_\Phi$. Hence, the transformation function f is learned by training both networks jointly on a parallel corpus containing data-points from the input and target distribution in an alternating way. The respective goal of these two networks is complimentary but opposite to each other. Adversarial networks have been recently used for speech enhancement [41], network representation learning [6], and medical image analysis [62]. However, their effectiveness in modeling context information for document information extraction remains unclear. We describe a machine-learning model based on an adversarial neural network that plays an important role in modeling context information in visually rich documents in Section 4.3.

D. Pretraining: To mitigate labeling effort to train our adversarial neural network on a new corpus, we train it on a different corpus with readily available, large-scale labeled data. Once trained, we initialize the network with parameters obtained from this process and continue training it on our target corpus [40]. We show in Section 5 that following this approach we can train a network on a minimal labeled samples and obtain satisfactory extraction performance.

E. Long Short-Term Memory Network: The machine-learning model we utilize in this paper to encode the local context of a visual span is based on a Long Short Term Memory (LSTM) network [1]. It is a type of recurrent neural network (RNN). RNN's take sequential data $X = \{x_1, x_2..., x_t\}$ as input. For each element in X, previous inputs can affect the output for the current element. The structure of an RNN is formally described as follows.

$$h_t = F(x_t, h_{t-1}) \tag{1}$$

$$y = G(\{h_1, h_2..., h_t\}) \tag{2}$$

Here, h_t represents the hidden state updated by the element x_t, and y denotes the sequence of hidden states generated till that timestep. The functions F and G are nonlinear transformations. For RNN's, F is defined as follows, $F = tanh(W_h x_t + U_h h_{t-1} + b_h)$, where W_h, U_h, and b_h are all learnable parameter matrices. The function G is typically specific to the downstream IE task. LSTM networks are a type of RNN that introduces a new structure called *gates* to learn long-term dependencies between different elements in the input sequence. An LSTM network typically has three types of gates: *input-gates* (I_t) that decide which values are updated in a memory cell, *forget-gates* (F_t) that decide which values stay in the memory, and *output-gates* (O_t) that decide upon the values in memory that going to be used to compute the output of the cell. The final structure of an LSTM network is defined as follows.

$$I_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$

$$F_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$

$$O_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

$$C_t = F_t \circ C_{t-1} + I_t \circ tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$

$$h_t = O_t \circ tanh(C_t) \tag{7}$$

C_t denotes the state vector of the memory cell, W, U, and b denote parameter matrices, σ denotes the sigmoid function, and $\circ$ denotes Hadamard product. A bidirectional LSTM network consists of a forward-LSTM and a backward LSTM network. The forward-LSTM network reads an input sequence $X = \{x_1, x_2..., x_t\}$ from x_1 to x_t and updates its hidden sequence h^f, whereas the backward-LSTM network reads the input sequence backward from x_t to x_1 and updates its hidden sequence h^b. The hidden vector representation of the bidirectional LSTM network is a concatenation of the forward and backward hidden sequences $[h^f, h^b]$.

F. Attention: To represent a potentially long input sequence without squashing it to a fixed-length internal representation and losing important information, the attention mechanism leverages a soft word-selection process conditioned on the global information of the input sequence [1]. Instead of truncating or pooling information from long sequences, this mechanism allows an LSTM network to pay more attention to a subset of elements in the input sequence

where the most relevant information is concentrated. We develop a multimodal bidirectional LSTM network with attention to encode the local context information of a visual span in Section 4.3.

4 THE ARTEMIS FRAMEWORK

`Artemis` works in two phases. In the first phase, it identifies a set of candidate visual spans for every named entity to be extracted. Each of these candidate spans are then searched to identify the smallest visual spans within them containing a named entity in the second phase. The key enabler of both phases is a data model that represents heterogeneous documents in a principled way. We describe it below.

4.1 `Artemis`' Data Model

We represent a document as a nested tuple (V, T), where V denotes the set of *atomic elements* in the document, and T denotes their visual organization. We describe both of them below.

4.1.1 *Atomic element*

An atomic element refers to the smallest visual element in a visually rich document. There can be two types of atomic elements in a document: *text element* and *image element*.

A. Text element: It is the smallest visual element in a document with a text-attribute. Assuming a rectangular coordinate system with the origin at the left-top corner, we can represent a text element a_t as a nested tuple $(text\text{-}data, x, y, w, h)$. Here, *text-data* represents the transcription of the visual span covered by a_t, h & w denote the height and width of the smallest bounding-box enclosing a_t. x & y represent coordinates of the top-left corner of the bounding-box. We transcribe a visual span using Tesseract [54], a popular open-source transcription engine. We deem each word as a text element in our data model.

B. Image element: It denotes an image-attribute in the document. We represent an image element a_i as a nested tuple $(image\text{-}data, x, y, w, h)$. Here, *image-data* represents the pixel-map, h & w denote the height and width, and x & y represent coordinates of the top-left corner of the smallest bounding box that encloses a_i.

4.1.2 *Visual organization*

We represent the visual organization of a document using a tree-like structure (T). Each node in T represents a visual span at various levels of layout hierarchy. Following the hOCR specification format by Breuel et al. [2], we define T with five levels of hierarchy. Each level corresponds to a specific level of abstraction. We define the hierarchy as follows: each document is made up of several *columns*, every *column* is divided into some *paragraphs*, every *paragraph* is split into *text-lines*, and every *text-line* consists of multiple *words*. We represent T by leveraging this hierarchy. A node (v_1) is a child of another node (v_2) in T if the visual span represented by v_1 is enclosed by the span represented by v_2. Leaf nodes of T represent individual words, whereas the root node represents the visual area covered by the single page of the document. We use an open-source page segmentation algorithm [54] to construct the layout-tree of a document.[1]

Our data model helps represent documents with diverse layouts and formats in a principled way. Leveraging the structure presented above, we can represent any visual span in the document as a nested tuple (v, t), where $v \subset V$ denotes the set of atomic elements appearing within the visual span and t denotes the smallest sub-tree of the document layout-tree T that contains all the atomic elements in v.

[1] we construct the layout-tree of a document from its rendered image rather than utilizing DOM tree-structures as it helps us to extend our method to document formats that do not support DOM-tree specifications

4.2 Candidate Visual Span Generation

`Artemis` relies on a number of weakly supervised functions to introduce domain-specific knowledge for identifying the candidate visual spans where a named entity may appear. We implement these as Python functions that accept a named entity, a list of mentions and where they appeared in the training corpus, and the test document D. Each function returns a set of candidate visual spans in D. These functions can range from simple regular expressions, publicly available libraries [13, 14] to complex functions that account for signals across both visual and textual modality. We provide examples of some of the functions used in our experiments below.

```
# WSF1: Approximate string matching based candidate generation
    def text_matcher(ne_lst,D,T){
        candidate_span_lst = []
        text = transcribe(D)
        for ne in ne_lst:
            if ne in text:
                span_coords = T.lookup(approx_match(text,ne))
                candidate_span_lst.append(span_coords)
        return candidate_span_lst
    }
```

Example 4.1: `WSF1` follows an approximate string matching approach to identify candidate visual spans in a document. For each named entity $n_i \in N$, it constructs a dictionary of all the entity mentions of n_i from the training corpus and finds a match in the transcribed text from D. We consider a phrase p in the transcribed text to be a match to an entity mention ne, if (a) they contain the same number of words, and (b) the minimum edit-distance between all word-pairs in p and ne is less than or equal to 3. If p is deemed a match, we look up the layout-tree T of the test document D and return the smallest visual span where p appears.

```
# WSF2: Parts-of-speech based candidate generation
    def parts_of_speech_tag_matcher(ne_lst,D,T){
        candidate_span_lst = []
        text = transcribe(D)
        tagged_text = tagger(text)
        for ne in ne_lst:
            seq = pos_tagger(ne)
            span_coords = T.lookup(match(tag_text,seq))
            candidate_span_lst.append(span_coords)
        return candidate_span_lst
    }
```

Example 4.2: `WSF2` utilizes a publicly available parts-of-speech tagger [13] to construct unique tag sequences representing a named entity n_i from its mentions in the training corpus. A phrase p appearing in the transcribed text of a test document D is deemed a match if the tag sequence obtained from p appears in the set of tag sequences representing n_i. Once a match is found, we look up

the smallest visual span where p appears in D from its document layout-tree and return it as a candidate visual span.

```
# WSF3: Visual positioning based candidate generation
    def position_matcher(ne_pos_lst,D,T){
        candidate_span_lst = []
        for ne_pos in pos_lst:
            text_line_coords = T.traverse(ne_pos)
            span_coords = pad(text_line_coords,50)
            candidate_span_lst.append(span_coords)
        return candidate_span_lst
    }
```

Example 4.3: Finally, the function WSF3 takes the visual modality of a document into account to identify candidate visual spans. Given a named entity n_i and a list of where each of its mentions appears in a training document, it traverses up the layout-tree of the training document to identify the text-lines (see Section 4.1) where the mention appears, dilates [42] the visual span enclosing those text-lines by 50 pixels (along both dimensions) and returns the resulting visual span.

The main objective of these functions in our extraction workflow is to identify candidate visual spans that are likely to contain a named entity $n_i \in N, \forall i$ with high-recall. Using a combination of these functions, we were able to identify candidate visual spans that contained true mentions of the named entities $n_i \in N$ to be extracted with an average recall-score[2] ranging from 92.75% to 97.50% on our datasets (see Fig. 3). Identifying the smallest visual span within each candidate span is the responsibility of the second phase of our workflow.

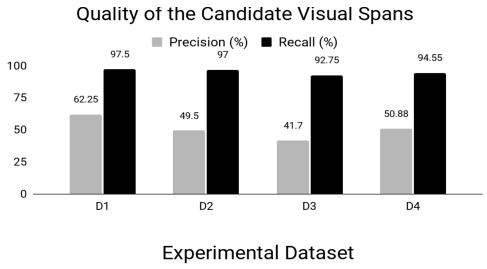


Figure 3: Weakly supervised functions identify visual spans containing true entity mentions with high recall

4.3 Identifying Local Context Boundaries

We represent a visual span using two fixed-length vectors β_L and β_G. β_L denotes the local context vector and β_G denotes the global context vector of the visual span. To compute β_L, we need to identify the local context boundary of the visual span first. Formally, given a visual span v in document D, the local context boundary of v is the smallest, semantically coherent visual span L in D that contains v. We decompose this task into two sub-problems. First, find a partition of D i.e., a set of non-overlapping visual spans $\{L_1, L_2 ... L_p\}$ in D that are isolated from each other by explicit or implicit visual cues.

[2]percentage of true entity mentions contained within the candidate visual spans

Figure 4: Local context boundary of a visual span appearing in a document (resized for illustration)

Second, find the visual span from this partition that contains the visual span v from the document layout-tree (Section 4.1). The key insight here is to utilize visual cues as delimiters of coherent semantic concepts to identify a visual span's local context. The local context boundary of a visual span within the solid rectangle in Fig. 4 (left) is shown within the dotted rectangle on the right.

One of the main challenges in developing a robust approach to find such a partition from a visually rich document is the inherent heterogeneity of real-world documents. Visual cues used as concept boundaries are often domain-specific and do not translate well to documents belonging to other domains. For example, the vertical space between two text-lines works as a visual delimiter in a tax-form document (see (A) in Fig. 1), but the same cannot be said for a news-article (see (C) in Fig. 1). Domain-agnostic visual segmentation algorithms such as [51] often do not find the optimal partition in such scenario. Layout-specific or rule-based approaches [27] are hard to scale. Supervised approaches, on the other hand, require large-scale labeled data that are prohibitively expensive to prepare. To incorporate domain-specific knowledge when finding the optimal partition without extensive feature engineering, we develop a deep neural network. We mitigate the requirement of human-labeled data to train this network by pretraining it on the segmentation output of a domain-agnostic, visual segmentation algorithm [51] on a large-scale publicly available dataset [15]. Once training converges, we continue training this network on a minimal training-set from our target dataset. Our network consists of a pair of convolutional networks, the Generator network, and the Discriminator network. We identified the architecture of both networks empirically.

4.3.1 *The Generator Network:*

The Generator network is a deep convolutional network [23]. Its primary objective is to learn a transformation function f_g that computes a segmentation mask of a visually rich document. During inference, it takes a squared[3] (512×512), rendered image of document D as input, and outputs a segmentation mask v_D (see Fig. 6). Physically, v_D is a binary image with the same dimensions as D, where each bounding-box represents the local context boundary of the visual spans that appear within that bounding-box.

[3]we square a rendered document image with dimensions $h \times w$, where $w \leq h$, by padding both sides of its smaller dimension with $\frac{h-w}{2}$ pixels of same color as the original boundary pixels in LAB colorspace

Architecture: The Generator network consists of a number of *encoder* and *decoder* blocks. Upon input, a document is propagated through the encoder blocks. Every encoder block (EN1-EN9 in Fig. 5) progressively down-samples its input from the previous block until the bottleneck-layer (a one-dimensional vector of length 16), which feeds into a decoder block. Each decoder block (DEC1-DEC9 in Fig. 5) progressively up-samples its input than the previous block. An encoder block consists of a convolution layer followed by batch-normalization and a rectified linear unit [23]. A decoder block consists of a deconvolution layer followed by batch normalization and a rectified linear unit. To address the vanishing gradient problem during training, skip connections are introduced [17] between every $i^{th}, 1 \leq i \leq 9$ pair of encoder and decoder blocks. We discuss the training of this network in Section 4.3.3. A detailed description of the network architecture is presented in Appendix A.

4.3.2 *The Discriminator Network:*

The Discriminator network is a deep convolutional network [23] whose primary objective is to learn a function f_d that validates the segmentation mask v_D constructed by the Generator network. During inference, the input to this network is a squared (512×512), rendered image of the input document (D) and the segmentation mask (v_D) constructed by the Generator network. It outputs a binary number based on its belief in the validity of the segmentation mask. If the output is 1, v_D is deemed valid, and we have obtained an optimal partition of the input document D.

Architecture: The Discriminator network consists of five *discriminator* blocks (Disc1-Disc5 in Fig. 5). Each discriminator block consists of a convolution layer followed by a rectified linear unit. Once the concatenated image is introduced to the first discriminator block, the network progressively down-samples it until the final discriminator block, which computes a *validity-matrix*. It is a 16×16 binary matrix. Each entry of this matrix represents the network's belief on the validity (0/1) of a patch in the learned segmentation mask. The validity of the segmentation mask is inferred by performing a point-wise convolution (i.e., filter-size = 1×1) on the validity-matrix and feeding the output to a sigmoid activation function. If the output of the activation function is non-zero, v_D is deemed valid. A more detailed overview of the network architecture is presented in Appendix A.

4.3.3 *Joint adversarial training:*

We pretrain both the Generator and Discriminator network from scratch with an adversarial learning objective [12] on the outputs of an unsupervised visual segmentation algorithm [51] on a large-scale publicly available dataset [15]. It is a domain-agnostic algorithm that accepts a rendered image of a document as input and returns its segmentation mask based on a number of commonly used visual delimiters defined beforehand. We construct a large-scale training corpus by feeding each document in this dataset to the segmentation algorithm [50] and saving the segmentation mask it outputs. Then, we randomly sample 20% of the image-pairs from this training corpus and introduce synthetic noise in them. Specifically, we use rotation based deformation and smooth deformations with random displacements. Once the training converges on this augmented corpus, we continue training our networks on a limited corpus from our target dataset by sampling $n = 15$ documents

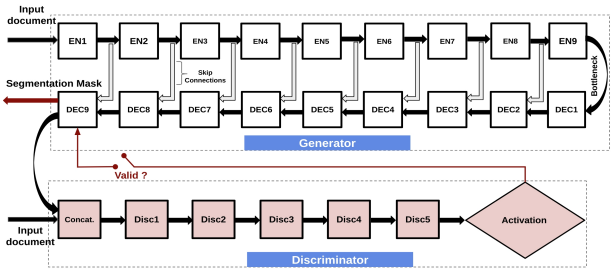


Figure 5: A snapshot of the Generator and Discriminator network during inference

of each type. This corpus contains a gold-standard segmentation mask for each sampled document. We describe the process of constructing these gold-standard annotations in Section 5. During both phases of the training procedure, the input to the network-pair is a pair of images – a squared, rendered image of the training document and its corresponding segmentation mask. Retraining the network-pair on human-labeled data helps incorporate knowledge specific to the target corpus during the construction of a segmentation mask. Through rigorous experiments (see Section 5), we will show that following this pretraining method provides satisfactory end-to-end performance on four cross-domain datasets without requiring large-scale human-labeled data.

Training objective: We train both networks jointly with an adversarial learning objective using alternating backpropagation [12], i.e., training alternates between a gradient descent step on the discriminator network and a step on the generator network until convergence. We used minibatch stochastic gradient descent for training and Adam-solver [20] for parameter optimization. The joint training objective of the network-pair $\mathcal{L}(G, F)$ is defined as follows.

$$\mathcal{L}(G, F) = L_{disc}(F) + L_{gen}(G) \tag{8}$$

$$L_{disc}(F) = \mathbb{E}_{x,y}[log(F(x, y))] \tag{9}$$

$$L_{gen}(G) = \mathbb{E}_{x,z}[log(1 - F(x, G(x, z)))] + \alpha L_1 \tag{10}$$

$$L_1 = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \tag{11}$$

In Eq. 8, $L_{gen}(G)$ and $L_{disc}(F)$ represent the learning objective of the Generator and Discriminator network, respectively. At each training step, the Generator network tries to minimize $L_{gen}(G)$, whereas the Discriminator network tries to maximize $L_{disc}(F)$. Training converges at the optimal value $\mathcal{L}^* = argmin_G max_F \mathcal{L}(G, F)$. We train both networks on an image-pair. The Generator network is trained to construct the gold-standard segmentation mask (x) from the input document (z). The Discriminator network, on the other hand, is trained to validate the segmentation mask (y in Eq. 9) constructed by the Generator network against the gold-standard segmentation mask (x). $L_{disc}(F)$ (Eq. 9) and the first term of $L_{gen}(G)$ (Eq. 10) represent a cross-entropy based loss function [23]. The second term of $L_{gen}(G)$, on the other hand, minimizes the L1 distance (Eq. 11) between the segmentation mask constructed by the Generator network and its corresponding gold-standard annotation. We set the value of α in Eq. 10 to 0.7 for all experiments.

4.3.4 *Inference.*

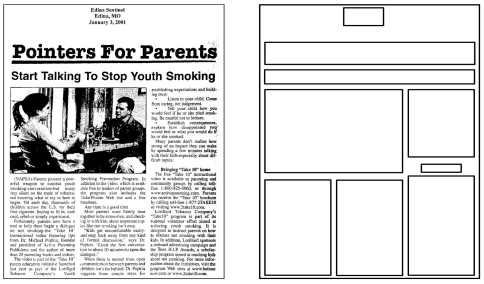
Once training converges, we feed a squared, rendered image of a test document (D) to the Generator network. It constructs a segmentation mask (v_D) and propagates it to the Discriminator network. If v_D is deemed valid, we can now proceed to determine the local context boundary of a visual span (v) in the document. Let, $\{L_1, L_2..., L_p\}$ denote the partitions found in the segmentation mask v_D. To identify the local context boundary of a visual span v, we look up the document layout-tree to find partitions $L_i \in v_D$ that contain at least one atomic element in v. If v spans across multiple partitions, we define its local context boundary as a set of all visual partitions in v_D that contains an atomic element in v. By developing a convolutional network that adaptively learns to identify the visual cues that act as a concept boundary in a visually rich document by taking domain-specific knowledge into account, we obviate the necessity of extensive feature engineering in our workflow. Learning the visual delimiters directly from the rendered image without relying on format-specific operators (e.g., structural tags in HTML documents) makes this process robust towards diverse document types. We compare our deep-learning-based approach to identify local context boundaries against several baseline methods, including a state-of-the-art neural network in Section 5. We obtained the best end-to-end performance in most cases using our adversarial network-pair. To identify the smallest visual span that contains a named entity, we encode its contextual information into a fixed-length vector. We describe this in the following section.

4.4 Filtering and Named Entity Recognition

For each named entity $n_i \in N$, we transcribe a candidate visual span v, convert the text to lowercase, normalize it, remove stopwords and chunk it into coherent text-segments using a publicly available natural language processing tool [34]. We formulate the task of identifying a mention of the named entity n_i within a candidate visual span v as a binary classification task. Let, c_{ij} denotes the j^{th} chunk obtained from the candidate visual span v for named entity n_i. We deem c_{ij} as an entity mention of n_i if it is assigned a 'True' label and return the smallest visual span enclosing c_{ij}. Otherwise, it is discarded. Let, $L = \{L_1, L_2...L_p\}$ denotes the local context boundary of the chunk c_{ij}, where $L_t, \forall t \in [1, p]$ represents a distinct partition in the learned segmentation mask of the input document containing at least one word in c_{ij}. We compute the probability of c_{ij} being inferred as a true entity mention of n_i as follows.

$$\Pi_{ij} = \Sigma_t^p P(b_c = 1 \mid b_{L_t} = 1)\,(P(b_c = 1 \mid b_D = 1) \tag{12}$$

In Eq. 12, the random variables b_c, b_{L_t}, and b_D represent the indicator variables denoting whether a true mention of the named entity n_i appears in the visual span containing c_{ij}, partition $L_t \in L$, and the document D respectively. To compute the first probability term in Eq. 12, we represent c_{ij} as a feature vector β_L, referred to as the *local context vector*. It encodes visual and textual semantics of c_{ij} from the visual span represented by L_t in the input document. The second probability term incorporates document-level context. It accounts for the fact that certain named entities are more likely to appear in one document type than another and encodes this information in a feature vector β_G, referred to as the *global context vector*. We discuss how both context vectors are computed below.



(a) Input document (b) Learned segmentation mask

Figure 6: (b) shows the segmentation masks constructed by the Generator network (v_D) for the document shown in (a)

4.4.1 *Encoding the local context vector.*

We extend a bidirectional LSTM network – the de-facto deep-learning standard [33] for natural language processing tasks with a set of dynamically generated visual features to model the local context information of c_{ij}. An overview of our network is shown in Fig. 7. In Section 5, we perform an exhaustive ablation study to evaluate the contribution of each modality on our end-to-end performance. We describe how the feature vectors are computed for each modality next.

Textual features: We compute textual features (h_{ijk}) of a word w_{ijk} in the chunk $c_{ij}, \forall j$ using a forward (represented by superscript f) and backward (represented by superscript b) LSTM network, summarizing the information about the whole chunk with a focus on the word w_{ijk}. It takes the following structure.

$$h^f_{ijk} = LSTM(h^f_{ij(k-1)}, e_{ijk}) \tag{13}$$

$$h^b_{ik} = LSTM(h^b_{ij(k-1)}, e_{ijk}) \tag{14}$$

$$h_{ijk} = [h^f_{ijk}, h^b_{ijk}] \tag{15}$$

In Eq. 13 and Eq. 14, e_{ijk} represents the word-embedding [57] of w_{ijk}, encoding its semantics into a fixed-length vector. We obtain the word-embedding w_{ijk} of each word using BERT-base [8], a state-of-the-art language model pretrained on English Wikipedia text. The contextualized feature representation of c_{ij}, denoted by t_{ij} is computed by following the attention mechanism to model the importance of different words that appear in c_{ij} and aggregate their feature representations as follows.

$$U_{ijk} = tanh(W_t h_{ijk} + b_t) \tag{16}$$

$$\alpha_{ijk} = \frac{exp(U^T_{ijk} u_t)}{\Sigma_m exp(U^T_{ijm} u_t)} \tag{17}$$

$$t_{ij} = \Sigma_m \alpha_{ijm} U_{ijm} \tag{18}$$

Here, W_t, u_t, and b_t are all learnable parameter matrices. U_{ijk} denotes a hidden representation of h_{ijk}, and α_{ijk} denotes the importance of the word $w_{ijk}, \forall k$. The final contextualized representation of c_{ij} is obtained by computing a weighted sum of $U_{ijk}, \forall k$.

Visual features: We compute a set of dynamic features (V_{ijk}) to encode the local visual context of each word in c_{ij}. A detailed

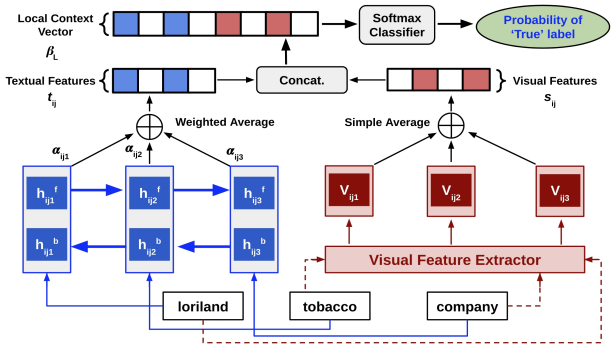


Figure 7: An overview of the Multimodal LSTM network

description of these visual features is presented in Appendix A. The key enabler behind these features is the learned segmentation mask returned by our adversarial neural network and the layout-tree of the input document. We identify the local context boundary of a word w_{ijk} in c_{ij} by traversing the layout-tree to find the partition in the segmentation mask that contains that word. These visual features encode many highly predictive semantic information implicitly, which we will show (in Section 5) compliments the supervisory signals gathered from the textual modality. We compute the final representation (s_{ij}) of c_{ij} by averaging the visual features obtained from each word in c_{ij} i.e., $s_{ij} = \frac{1}{K}\Sigma_k V_{ijk}$. The local context vector (β_L) is therefore obtained by concatenating the textual feature vector t_{ij} and the visual feature vector s_{ij}, i.e. $\beta_L = [t_{ij}, s_{ij}]$. The final layer of our network is a fully-connected layer that takes the feature vector β_L as input and outputs the probability of c_{ij} being a true mention of the named entity n_i.

Training: All network parameters are learned following a joint training paradigm. This includes the parameter matrices in the bidirectional LSTM network as well as the weights of the last softmax layer. For each named entity $n_i \in N$, we train the network on contextualized vectors computed from the entity mentions of n_i in our training corpus. Given the position of an entity mention in a training document, we identify its local context boundary by leveraging the segmentation mask returned by our pretrained adversarial neural network. We use a cross-entropy based learning objective [23] for training.

Inference: Once training converges, we feed each chunk c_{ij} obtained from a candidate visual span v for named entity n_i in the test document. The softmax layer outputs the probability of c_{ij} being a true mention of the named entity n_i. This represents the first probability term i.e., $P(b_c = 1 \mid b_{L_t} = 1)$ in Eq. 12.

4.4.2 *Encoding the global context vector:*

The global context vector β_G of a visual span is a fixed-length, discriminative feature-vector representing the input document D in which it appears. It accounts for the fact that mentions of certain named entity types are more likely to appear in certain document types than others. For instance, a mention of the named entity "Side effects" is more likely to appear in a news article on tobacco addiction than a real-estate flyer. We incorporate this intuition using corpus-level statistics to infer the probability of a visual span containing a true mention of the named entity $n_i \in N$ in D. We compute the global context vector β_G using a state-of-art, deep convolutional network [50] pretrained on a large-scale publicly available dataset [15]. Once training converges, we feed a rendered image of our input document to this network and obtain the 512-dimensional vector from the final fully-connected layer of this network.

Inference: Following this process for all documents in the training corpus, we group the training documents into a number of non-overlapping clusters using the DBSCAN algorithm [53]. For each cluster, we compute the fraction of documents that contain a mention of the named entity $n_i \in N$. Let, p_j denote the fraction of documents in the j^{th} cluster that contain a mention of n_i, and c_j denotes the feature representation of the j^{th} cluster-center (average of the context vectors in each cluster). Let, d_j denotes the Euclidean distance between c_j and the global context vector representation of the test document. We compute the probability of a true mention of the named entity n_i appearing in the test document as follows.

$$P(b_c = 1 \mid b_D = 1) = \phi_{t^*} p_{t^*} \tag{19}$$

$$\phi_{t^*} = \frac{exp(d_{t^*})}{\Sigma_t exp(d_t)} \tag{20}$$

$$t^* = argmin_t(d_t) \tag{21}$$

Eq. 19 represents the second probability term of Eq. 12. Combining it with the softmax probability obtained from our LSTM network helps compute the probability term Π_{ij} – the probability of a chunk c_{ij} being a true mention of the named entity $n_i \in N$. We repeat this process for every chunk c_{ij} from all candidate visual spans. If the $\Pi_{ij} > 0.5$, we return the smallest visual span containing c_{ij}. We evaluate our end-to-end performance on a suite of IE tasks from heterogeneous, visually rich documents in Section 5.

5 EXPERIMENTS

We evaluate `Artemis` on four cross-domain datasets in our experiments, the NIST Tax dataset (D1), the MARG dataset (D2), the Tobacco Litigation dataset (D3), and the Brains dataset (D4). Documents in each of these datasets are heterogeneous in nature, i.e., they either originate from different sources or have different layouts or formats. We seek to answer three key questions in our experiments: (a) how did we perform on each dataset? (b) how did it compare against text-based methods? and (c) what were the individual contributions of various components in our workflow? We answer the first two questions in Sections 5.3.1. To answer the last question, we perform an ablative analysis in Sections 5.3.2 and 5.3.3. We used Keras [19] to implement our neural networks. All experiments were performed with 1TB RAM and a 12GB Titan-XP GPU.

5.1 Datasets

To evaluate the generalizability of our method, we measure its performance on four datasets for separate IE tasks. Each task required identifying the smallest visual spans containing true mentions of a list of named entities in a single-page document. A detailed list of these named entities and a short description of what they represent are available at: https://github.com/sarkhelritesh/ewr. The datasets D1, D2, and D3 are publicly available and have been used by previous works [51]; we prepared dataset D4 for this study. In

Table 1: Average top-1 accuracy and F1-score obtained by all competing methods on our experimental datasets

Index	Dataset	Text-only (A1)		ReportMiner (A2)		Graph-based (A3)		Weak Supervision (A4)		Artemis	
		Accuracy (%)	*F1* (%)	*Accuracy* (%)	*F1* (%)	*Accuracy* (%)	*F1* (%)	*Accuracy* (%)	*F1* (%)	*Accuracy* (%)	*F1* (%)
D1	NIST Dataset	89.75	86.33	**97.50**	**93.25**	95.50	91.86	95.50	92.0	95.55	92.60
D2	MARG Dataset	69.45	67.50	67.70	62.25	72.0	70.95	71.25	69.07	**74.33**	**72.50**
D3	Tobacco Litigation Dataset	51.20	49.65	59.70	55.25	65.25	62.90	63.50	61.35	**68.50**	**67.25**
D4	Brains Dataset	68.50	64.33	62.07	56.50	74.25	70.96	74.50	69.42	**78.40**	**74.35**

the following sections, we describe some of the key characteristics and the IE tasks defined on each dataset.

A. NIST Dataset (D1): The NIST special dataset-6 [38] is a publicly available dataset containing 5595 scanned documents representing 20 different forms from the IRS-1040 package. We defined an IE task with 1369 unique named entities (NE) on this dataset. Examples include form-fields such as 'Taxpayer's name', 'Address', 'Name of the dependents', and 'Gross income'.

B. MARG Dataset (D2): The Medical Article Record Groundtruth (MARG) dataset [56] is a publicly available collection of biomedical journals from the National Library of Medicine. It contains 1553 articles with nine different templates. Documents in this dataset exhibit high variance in intra-class layout similarity. We defined an IE task on this dataset with four unique named entities, 'Article title', 'Contribution', 'Authors', and 'Affiliation'.

C. Tobacco Litigation Dataset (D3): The Tobacco Litigation dataset [24] is a benchmark dataset containing 3482 documents from publicly available litigation records against US tobacco companies in 1998. It contains ten commonly used document types, including email and newspaper article. Documents in this dataset are noisy (e.g., low contrast, broken characters, salt-and-pepper noise), and exhibit high intra-class and low inter-class variance in layout similarity, making it one of the most challenging datasets in our experimental setup. We defined an IE task with 34 unique named entities on this dataset. For example, the named entities defined for an email document included 'Sender's name', 'Receivers' name', and ' Receivers' affiliations'. A complete list of named entities defined for this task can be found in the supplementary document.

D. Brains Dataset (D4): In the context of emergency care [36], a 'Brain' document refers to a physical or digital document used by registered nurses as a record-keeping tool to maintain vital information of a patient under care. Information from this document is manually added to a centralized database periodically to update it. A 'Brain' document typically utilizes a number of visual cues to identify vital information quickly [52]. We prepared a synthetic dataset from 36 publicly available 'Brain' templates [16] used by registered nurses in the United States. We populate each document with artificial entries mimicking real-world data [36]. The dataset contains approximately 1M documents. We defined an IE task on this dataset with 12 distinct named entities, typically used as patient identifiers [52], such as 'Name', 'Age', 'Medical history'. Sample documents from each of our four datasets are shown in Fig. 1.

5.2 Experiment Design

We closely follow the evaluation scheme proposed in [51]. A visual span v_i inferred as a mention of the named entity n_i by our IE workflow is deemed to be accurate if: (a) it is accurately localized, i.e. its position aligns with gold-standard coordinates, and (b) it is accurately classified, i.e. n_i is the gold-standard label assigned to v_i.

5.2.1 *Groundtruth construction:*

Following the guidelines proposed by Clavelli et al. [5], we construct multi-level groundtruth representation for each document. We annotate each document at both word-level and local context boundary-level using a publicly available annotation tool. Three graduate students, each familiar with at least one of the datasets, were invited to help construct the groundtruth data. The list of named entities N for each IE task was decided by consensus. For each dataset, guidelines to identify visual delimiters that act as concept boundaries in the document were proposed by an expert familiar with the dataset first and then finalized after reaching cohort consensus. Once the guidelines were fixed, each annotator was asked to provide the following information for each document: (a) word-level bounding-boxes of the smallest visual span of an entity mention, (b) the named entity type associated with that mention, and (c) bounding-boxes of the smallest, visually isolated, coherent area in the document that contains that mention. Each document was annotated by at least two annotators. We obtained the gold-standard positional groundtruth by averaging the coordinates provided by each annotator, and the gold-standard entity-type by majority voting.

5.2.2 *Evaluation protocol:*

Suppose the visual span v_i is inferred as a mention of the named entity n_i. (x_i, y_i) denote the x,y coordinates of the left-top-most point of the smallest bounding box enclosing v_i, and h_i, w_i represent the height and width of this bounding-box. We evaluate the validity of this prediction by first checking if v_i is accurately localized. Following Everingham et al. [10], we consider v_i to be accurately localized if its intersection-over-union against the gold-standard annotation is greater than 0.65. If v_i is deemed to be accurately localized, we check if the groundtruth named entity type assigned to v_i is n_i. If it is, we consider it a true mention of the named entity n_i.

5.2.3 *Training and test corpus construction:*

We randomly sampled 60% of each document type to construct the training corpus for a dataset. The rest of the documents were partitioned in a 50:50 ratio to construct the test and validation set. The adversarial neural network was pretrained on the RVL-CDIP dataset [15] first and then fine-tuned [40] on a corpus consisting $n = 15$ documents randomly sampled from each document type from our experimental dataset. We trained our LSTM network on multimodal features extracted from each document in the training corpus. The segmentation mask used to identify the local context boundary of a document was constructed by an adversarial neural network fine-tuned on that dataset.

Table 2: Average top-1 accuracy and F1-score obtained by all baseline methods on our experimental datasets

Index	Dataset	Convolution (B1)		Fixed-Area (B2)		Unsupervised (B3)		Sentence-based (B4)		Artemis	
		Accuracy (%)	*F1 (%)*	*Accuracy (%)*	*F1 (%)*	*Accuracy (%)*	*F1 (%)*	*Accuracy (%)*	*F1 (%)*	*Accuracy (%)*	*F1 (%)*
D1	NIST Dataset	95.05	92.28	94.70	92.33	95.0	92.27	95.50	**92.55**	**95.55**	**92.60**
D2	MARG Dataset	72.65	68.25	73.50	**74.85**	**73.42**	69.50	68.70	65.25	74.33	72.50
D3	Tobacco Litigation Dataset	65.80	62.75	61.50	55.12	64.33	63.20	57.70	54.20	**68.50**	**67.25**
D4	Brains Dataset	75.42	72.25	72.15	70.0	73.25	71.82	69.50	66.55	**78.40**	**74.35**

5.3 Results and Discussion

We report the average top-1 accuracy and macro-F1 score for each dataset in Table 1. In general, we observed a positive correlation between our end-to-end performance and the discriminative properties [50] of a dataset. We performed better on datasets that are more visually discriminative.

5.3.1 End-to-end comparison against contemporary methods: We compare our end-to-end extraction performance against several contemporary IE methods. The average top-1 accuracy and F1-score obtained for each dataset are shown in Table 1. The best performance obtained for a dataset is boldfaced.

Our first baseline (A1) method takes a text-only approach for information extraction. It takes the rendered image of a document as input and transcribes it. For fair comparison, we used Tesseract, the same transcription engine [54] used in our extraction workflow. After cleaning the document (following the same preprocessing steps), the text is chunked [34] and each chunk is fed to BERT-base [8], a state-of-the-art neural model pretrained on Wikipedia corpus. It generates a fixed-length vector representation of that chunk. No explicit context information from the document is incorporated at any stage of this baseline. This vector is then fed to a bidirectional LSTM network with attention [30] to infer whether that chunk contains a true mention of a named entity defined for that dataset. We outperform this baseline on all datasets.

Our second baseline method (A2) is ReportMiner [27], a commercially available, human-in-the-loop, IE tool. It allows the users to define custom masks for every named entity to be extracted. We defined rules for every document type in our training corpus using this tool. Information extraction was performed by selecting the most appropriate rule and applying it on a test document based on layout similarity. This baseline method performed better on the fixed-format NIST dataset; however, performance degraded as the variability in document layouts increased. We were able to outperform this method on most datasets with significantly lesser human effort in our end-to-end workflow.

Our third baseline method (A3) follows a graph-based approach to model the relationship between different visual spans in the document. After decomposing the document using a domain-agnostic segmentation algorithm, it represents each text-element using a concatenation of feature vectors obtained from a graph-convolution network [26] and a pretrained language model. This concatenated vector is then fed to a bidirectional LSTM network, followed by a conditional random field (CRF) to identify the entity mentions. We outperformed this baseline on two of the most challenging datasets in our setup, D3, and D4. Performance was comparable on dataset D1.

Our final baseline (A4) method follows a data-programming paradigm similar to Fonduer [59]. It employs the weakly supervised functions introduced in Section 4.2 to identify candidate visual spans in a training document. These weakly supervised functions along with the noisy candidate spans are then passed to Snorkel [44], a data-programming engine [46]. It converts the noisy candidate spans to denoised labeled visual spans. Each visual span is then encoded into a fixed-length vector using the same multimodal feature library in [59] and fed to a bidirectional LSTM network. It is worth noting here that the HTML-specific features used in [59] could only be extracted from datasets D2 and D4 in our experimental settings as they contain PDF documents that could be converted to HTML format using an opensource tool[4] during preprocessing. Once trained, the LSTM network is then used to infer the named entities appearing within a candidate visual span in the test corpus. Although it requires less human-labeled data in its end-to-end workflow, we were able to outperform this weakly supervised baseline on three of our datasets. Performance was comparable on dataset D1.

Table 3: Improvement in F1-score over ablative baselines

ve

Index	Type	ΔF1(%)			
		D1	D2	D3	D4
S1	Textual features only	0.80	2.50	3.88	2.15
S2	Visual features only	5.45	25.40	45.22	37.50
S3	Local context only	1.06	0.0	2.50	1.95

5.3.2 Comparing various ways to identify local context boundary: In this section, we take a more in-depth look at various ways to identify the local context boundary of a visual span and report their end-to-end performance on our experimental datasets in Table 2. In the first baseline method (B1), we employ a state-of-the-art, fully convolutional network [47] trained similarly to our adversarial neural network to identify the local context boundary of a visual span. The rest of the workflow is kept similar to ours. In our second baseline method (B2), we identify an area of fixed dimensions around a visual span as its local context boundary. The rest of the workflow stays the same as ours. Our third baseline method (B3) utilizes an unsupervised, domain-agnostic visual segmentation algorithm [51] to identify the local context boundary of a visual span v. Once context boundary is determined, it follows the same extraction workflow. Our final baseline method (B4) identifies the local context boundary of a visual span as the sentence that contains it. Due to the heterogeneous nature of documents in our experimental datasets, we define the sentence containing a visual span as the non-empty set of text-lines that contain at least one atomic element

[4] https://poppler.freedesktop.org

appearing within the visual span. We perform comparably or better than all baseline methods. In general, extraction performance was good if context boundaries were determined adaptively, taking local, domain-specific knowledge into account. Improvement in performance over B4 reveals that capturing semantic information beyond sentence-level boundaries is important for extraction performance. The baseline B2 performed comparably on all datasets. However, we note that identifying the optimal size of the context window affects end-to-end extraction performance and may need to be carefully selected based on the visual span and the dataset.

5.3.3 Ablation study:
We investigate the individual contributions of textual and visual features used to encode the local context vector in our extraction workflow in scenarios S1 and S2 of Table 3. For both rows, the final column quantifies the contribution of a modality by measuring the improvement in F1-score obtained over the ablative baseline by our method. In S1, we only consider the textual features computed by the bidirectional LSTM network to encode the local context vector of a visual span. We observed the highest improvement in performance against this ablative baseline for dataset D3, which was closely followed by datasets D2 and D4. This establishes the critical role played by visual features in representing context information of a visual span. Considering only the visual features to encode the local context vector (S2), on the other hand, incurs a noticeable decrease in extraction performance on all datasets. Finally, in scenario S3 of Table 3, we disregard the contribution of the global context vector to infer whether a visual span contains a true entity mention. We observed the highest decrease in F1-score on dataset D3 using this ablative baseline. No effect in performance was observed for dataset D2 as all of the named entities defined for this dataset appeared in every document. Including corpus-level statistics, therefore, did not have any effect on our end-to-end extraction performance.

6 RELATED WORK

Layout driven methods: One of the earliest approaches for information extraction from web documents is wrapper induction [21, 32, 35]. In this approach, prior knowledge about the layout of a document is taken into account to define custom masks for localizing and extracting named entities. These methods, however, do not scale well for documents with diverse layouts. Although not for named entity recognition, some recent works [37, 50] have proposed methods for template-free parsing of visually rich documents. For example, Cai et al. [3] proposed VIPS, an unsupervised method to produce a segmentation mask of a webpage. Their method was recently outperformed by Sarkhel et al. [51]. They proposed a generalized segmentation algorithm that not only takes the visual features of a document into account but also the semantic similarity of neighboring visual spans. Davis et al. [7] proposed a fully convolutional network to pair handwritten entries with preprinted text-fields in form documents. A fully convolutional network was also utilized by Yang et al. [61] to infer the semantic structure of a document. We compared our end-to-end performance against a fully convolutional baseline. We outperformed it on multiple datasets.

Format driven methods: A significant amount of existing works on document information extraction leverage format-operators and structural tags supported by rich document markup languages. Researchers like Manabe et al. [32] and Sun et al. [55] utilized HTML-specific operators to propose heuristics-based IE methods from HTML documents. Gallo et al. utilized PDF-specific format-operators for information extraction from digital flyers in [11]. A distance supervision approach was proposed by Lockard et al. [28] for relation extraction from semi-structured web documents. XML and HTML-specific feature descriptors were also leveraged by Deepdive [39] to extract relational tuples from visually rich documents. Contrary to such works, we utilize format-agnostic features in our workflow, making it robust towards diverse document types.

Supervision sources: Existing literature on document information extraction is heavily biased towards using textual features as the only supervision source. Some recent works [18, 59] have proposed multimodal approaches for information extraction. In addition to semantic features, Wu et al. [59] also considered structural and tabular modalities to perform information extraction from richly formatted documents. Only supervision in their workflow was provided in the form of heuristics-based labeling functions. Following Fonduer's official source-code, we implemented a baseline that follows a similar weakly supervised approach in Section 5.3. We outperformed this baseline on three of our experimental datasets. Recently, Katti et al. [18] have proposed a supervised approach by taking positional information of individual characters in the document using a fully convolutional network. Although reasonable for sparse documents, it is hard to scale their approach for heterogeneous documents, especially documents that are textually dense. Human-in-the-loop workflows [27], crowd-sourcing [4], and domain expertise [59] have also been proven to be effective sources of weak supervision. These methods are, however, expensive to maintain and hard to scale. In Section 5.3, we compared our end-to-end performance against a commercially available, human-in-loop document IE tool [27]. We performed better than this baseline on most datasets with lesser human effort in our end-to-end workflow.

Generalized IE methods: Liu et al. [26] proposed a graph-based approach to encode multimodal context of visual spans for IE tasks. We outperformed this baseline on most tasks (see Table 1). In [51], Sarkhel et al. proposed a segmentation-based IE approach. After segmenting a document into a number of homogeneous visual areas, they employed a distance supervision approach to search for named entities within each visual area by scoring each candidate textual span in a multimodal encoding space. We have discussed the limitations of the domain-agnostic segmentation algorithm used in [51] in Section 4.2, and showed improvements over it in Table 2. In their recent work, Majumdar et al. [31] followed a similar approach. To identify a named entity in a business-invoice, they scored each text-element based on its similarity with the text-elements appearing within a fixed-sized window centered around it using a self-attentive neural network. Due to the homogeneous nature of their task, they were able to incorporate a number of task-specific features into their workflow. In `Artemis`, we develop a more robust framework and establish our efficacy on four separate IE tasks.

7 CONCLUSION

We have proposed an automated method for information extraction from heterogeneous visually rich documents in this paper. To identify the smallest visual span containing a named entity, we represent

each visual span using two fixed-length vectors. We develop a bidirectional LSTM network to encode the local multimodal context of a visual span. To incorporate domain-specific knowledge for identifying the local context boundaries, we develop an adversarial neural network based machine-learning model, trained with minimal human-labeled data. It determines the local context boundary of a visual span by taking local layout-specific cues into account. The feature vectors used in our end-to-end workflow are agnostic towards document layouts and formats which makes our method robust towards diverse document types. Experiments on four cross-domain datasets suggest that our method is generalizable and can outperform text-based IE methods on all datasets. In future, we would like to extend this work to nested documents and interactive workflows. The latter can potentially be achieved by constructing a representative summary of the document at controllable lengths [49]. We also plan to explore the dependency between different named entity types to derive a richer context model. Augmenting each extraction task with a human-interpretable explanation i.e., why a visual span is categorized as a certain named entity type is another exciting direction for domains such as healthcare informatics and insurance.

8 APPENDIX A

Table 4: Visual features extracted from a visual span. The second column denotes whether a feature is computed from the local context boundary of a word, or the visual span containing the word itself

Index	Visual span	Feature description
f_1	Local context boundary	Ranked order in a list of all partitions in the segmentation mask sorted based on average bounding-box height of text-elements appearing in it
f_2	Local context boundary	Number of text-lines
f_3	Local context boundary	Number of words
f_4	Local context boundary	Ranked order in a list of all partitions in the segmentation mask sorted based on surface area
f_5	Local context boundary	Ranked order in a list of all partitions in the segmentation mask topologically sorted on centroid-coordinates
f_6	Local context boundary	Height of the largest bounding-box enclosing a text-element in the visual span
f_7	Local context boundary	Euclidean distance from the nearest partition in the segmentation mask
f_8	Word	Height of the smallest bounding-box enclosing the visual span
f_9	Word	Relative coordinates of the centroid

Table 5: An overview of the Generator network architecture. 'K' denotes the dimension of each convolutional filter, and 'S' denotes the stride-length. The architecture of an encoder (`EN`) and a decoder (`DEC`) block is shown in Table 7

Operation Type	Symbol	Input	Operator	K	S
Encoder	EN1	$512^2 \times 3$	*encoder-block*	3	2
	EN2	$256^2 \times 2$	*encoder-block*	3	2
	EN3	$128^2 \times 4$	*encoder-block*	3	2
	EN4	$64^2 \times 8$	*encoder-block*	3	2
	EN5	$32^2 \times 16$	*encoder-block*	3	2
	EN6	$16^2 \times 16$	*encoder-block*	3	2
	EN7	$8^2 \times 16$	*encoder-block*	3	2
	EN8	$4^2 \times 16$	*encoder-block*	3	2
	EN9	$2^2 \times 16$	*encoder-block*	2	1
Decoder	DEC1	$1^2 \times 16$	*decoder-block*	2	1
	DEC2	$2^2 \times 32$	*decoder-block*	2	2
	DEC3	$4^2 \times 32$	*decoder-block*	2	2
	DEC4	$8^2 \times 32$	*decoder-block*	2	2
	DEC5	$16^2 \times 32$	*decoder-block*	2	2
	DEC6	$32^2 \times 32$	*decoder-block*	2	2
	DEC7	$64^2 \times 16$	*decoder-block*	2	2
	DEC8	$128^2 \times 8$	*decoder-block*	2	2
	DEC9	$256^2 \times 4$	*decoder-block*	2	2

Table 6: An overview of the Discriminator network architecture. 'K' denotes the dimension of each filter, and 'S' denotes the stride-length of in each encoder and decoder blocks. The architecture of each discriminator-block (`DISC`) is shown in Table 7

Block Type	Input	Operator	K	S
Input	$512^2 \times 3$+$512^2 \times 4$	*concatenation*	-	-
DISC1	$512^2 \times 7$	*discriminator-block*	2	2
DISC2	$256^2 \times 2$	*discriminator-block*	2	2
DISC3	$128^2 \times 4$	*discriminator-block*	2	2
DISC4	$64^2 \times 8$	*discriminator-block*	2	2
DISC5	$32^2 \times 16$	*discriminator-block*	2	2
Validity Matrix	$16^2 \times 1$	1×1 *convolution + sigmoid activation*	-	-

Table 7: Architecture of an encoder, decoder, and discriminator block. 'h' and 'w' denote dimensions of the input to each layer. 'c' denotes the number of input channels and c* denotes the output channel size. For each encoder and discriminator-block c* = c, for each decoder-block c* = 2c

Block	Input	Operator	Output
encoder-block	$h \times w \times c$	conv2d	$h \times w \times c$
	$h \times w \times c$	Batch Normalization	$h \times w \times c$
	$h \times w \times c$	ReLU	$h \times w \times c^*$
decoder-block	$h \times w \times c$	conv2d-transpose	$h \times w \times c$
	$h \times w \times c$	Batch Normalization	$h \times w \times c$
	$h \times w \times c$	Dropout + Skip concat.	$h \times w \times 2c$
	$h \times w \times 2c$	ReLU	$h \times w \times c^*$
discriminator-block	$h \times w \times c$	conv2d	$h \times w \times c$
	$h \times w \times c$	ReLU	$h \times w \times c^*$

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] T. Breuel. 2007. The hOCR Microformat for OCR Workflow and Results. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 1063–1067. https://doi.org/10.1109/ICDAR.2007.4377078

[3] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Vips: a vision-based page segmentation algorithm. (2003).

[4] Kuang Chen, Akshay Kannan, Yoriyasu Yano, Joseph M Hellerstein, and Tapan S Parikh. 2012. Shreddr: pipelined paper digitization for low-resource organizations. In *Proceedings of the 2nd ACM Symposium on Computing for Development*. 3.

[5] Antonio Clavelli, Dimosthenis Karatzas, and Josep Llados. 2010. A framework for the assessment of text extraction algorithms on complex colour images. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. 19–26.

[6] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial network embedding. In *Thirty-second AAAI conference on artificial intelligence*.

[7] Brian Davis, Bryan Morse, Scott Cohen, Brian Price, and Chris Tensmeyer. 2019. Deep visual template-free form parsing. In *2019 International Conference on*

Document Analysis and Recognition (ICDAR). IEEE, 134–141.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] AnHai Doan, Jeffrey F Naughton, Raghu Ramakrishnan, Akanksha Baid, Xiaoyong Chai, Fei Chen, Ting Chen, Eric Chu, Pedro DeRose, Byron Gao, et al. 2009. Information extraction challenges in managing unstructured data. *ACM SIGMOD Record* 37, 4 (2009), 14–20.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (VOC) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[11] Ignazio Gallo, Alessandro Zamberletti, and Lucia Noce. 2015. Content extraction from marketing flyers. In *International Conference on Computer Analysis of Images and Patterns*. Springer, 325–336.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[13] The Stanford NLP Group. 2020. *Stanford Part-Of-Speech Tagger*. Accessed: 2020-01-31.

[14] The Stanford NLP Group. 2020. *Stanford Word Tokenizer*. Accessed: 2020-01-31.

[15] Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. [n.d.]. Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*.

[16] Nurse Tech Inc. 2018. *NurseBrains*. Accessed: 2019-01-25.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. *arXiv preprint* (2017).

[18] Anoop Raveendra Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2d documents. *arXiv preprint arXiv:1809.08799* (2018).

[19] Keras. 2018. *Keras: Deep Learning for Humans*. Accessed: 2018-09-30.

[20] D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, Vol. 5.

[21] Nicholas Kushmerick. 2000. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence* 118, 1-2 (2000), 15–68.

[22] Matthew Lamm. 2020. *Natural Language Processing with Deep Learning*. Accessed: 2020-01-31.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[24] David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, D Grossman, and Jefferson Heard. 2006. Building a test collection for complex document information processing. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 665–666.

[25] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* (2015).

[26] Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 2019. Graph convolution for multimodal information extraction from visually rich documents. *arXiv preprint arXiv:1903.11279* (2019).

[27] Astera LLC. 2018. *ReportMiner: A Data Extraction Solution*. Accessed: 2018-09-30.

[28] Colin Lockard, Xin Luna Dong, Arash Einolghozati, and Prashant Shiralkar. 2018. Ceres: Distantly supervised relation extraction from the semi-structured web. *arXiv preprint arXiv:1804.04635* (2018).

[29] Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. *arXiv preprint arXiv:1904.03296* (2019).

[30] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* (2016).

[31] Bodhisattwa Prasad Majumder, Navneet Potti, Sandeep Tata, James Bradley Wendt, Qi Zhao, and Marc Najork. 2020. Representation learning for information extraction from form-like documents. In *proceedings of the 58th annual meeting of the Association for Computational Linguistics*. 6495–6504.

[32] Tomohiro Manabe and Keishi Tajima. 2015. Extracting logical hierarchical structure of HTML documents based on headings. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1606–1617.

[33] Christopher Manning. 2017. *Representations for language: From word embeddings to sentence meanings*. Accessed: 2020-01-31.

[34] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[35] Marcin Michał Mirończuk. 2018. The BigGrams: the semi-supervised information extraction system from HTML: an improvement in the wrapper induction. *Knowledge and Information Systems* 54, 3 (2018), 711–776.

[36] Austin F Mount-Campbell, Kevin D Evans, David D Woods, Esther M Chipps, Susan D Moffatt-Bruce, and Emily S Patterson. 2019. Value and usage of a workaround artifact: A cognitive work analysis of "brains" use by hospital nurses. *Journal of Cognitive Engineering and Decision Making* 13, 2 (2019), 67–80.

[37] Bastien Moysset, Christopher Kermorvant, Christian Wolf, and Jérôme Louradour. 2015. Paragraph text segmentation into lines with recurrent neural networks. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 456–460.

[38] NIST. 2018. *NIST Special Database 6*. Accessed: 2018-09-30.

[39] Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. 2012. DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference. *VLDS* 12 (2012), 25–28.

[40] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[41] Santiago Pascual, Antonio Bonafonte, and Joan Serra. 2017. SEGAN: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452* (2017).

[42] Frédéric Patin. 2003. An introduction to digital image processing. *online]: http://www. programmersheaven. com/articles/patin/ImageProc. pdf* (2003).

[43] P David Pearson, Michael L Kamil, Peter B Mosenthal, Rebecca Barr, et al. 2016. *Handbook of reading research*. Routledge.

[44] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.

[45] Alexander J Ratner, Stephen H Bach, Henry R Ehrenberg, and Chris Ré. 2017. Snorkel: Fast training set generation for information extraction. In *Proceedings of the 2017 ACM international conference on management of data*. 1683–1686.

[46] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*. 3567–3575.

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[48] Sunita Sarawagi et al. 2008. Information extraction. *Foundations and Trends® in Databases* 1, 3 (2008), 261–377.

[49] Ritesh Sarkhel, Moniba Keymanesh, Arnab Nandi, and Srinivasan Parthasarathy. 2020. Interpretable Multi-headed Attention for Abstractive Summarization at Controllable Lengths. In *Proceedings of the 28th International Conference on Computational Linguistics*. 6871–6882.

[50] Ritesh Sarkhel and Arnab Nandi. 2019. Deterministic routing between layout abstractions for multi-scale classification of visually rich documents. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 3360–3366.

[51] Ritesh Sarkhel and Arnab Nandi. 2019. Visual segmentation for information extraction from heterogeneous visually rich documents. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 247–262.

[52] Ritesh Sarkhel, Jacob J Socha, Austin Mount-Campbell, Susan Moffatt-Bruce, Simon Fernandez, Kashvi Patel, Arnab Nandi, and Emily S Patterson. 2018. How Nurses Identify Hospitalized Patients on Their Personal Notes: Findings From Analyzing 'Brains' Headers with Multiple Raters. In *Proceedings of the International Symposium on Human Factors and Ergonomics in Health Care*, Vol. 7. SAGE Publications Sage India: New Delhi, India, 205–209.

[53] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.

[54] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, Vol. 2. IEEE, 629–633.

[55] Fei Sun, Dandan Song, and Lejian Liao. 2011. Dom based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 245–254.

[56] GFG Thoma. 2003. Ground truth data for document image analysis. In *Symposium on document image understanding and technology (SDIUT)*. 199–205.

[57] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. 384–394.

[58] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546* (2019).

[59] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1301–1316.

[60] Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. *arXiv preprint arXiv:1609.03632* (2016).

[61] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. 2017. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5315–5324.

[62] Xin Yi, Ekta Walia, and Paul Babyn. 2019. Generative adversarial network in medical imaging: A review. *Medical image analysis* 58 (2019), 101552.