Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel

Yao-Hung Hubert Tsai[1] **Shaojie Bai**[1] **Makoto Yamada**[34]
Louis-Philippe Morency[2] **Ruslan Salakhutdinov**[1]
{[1]Machine Learning Department,[2]Language Technology Institute}, Carnegie Mellon University
[3]Kyoto University [4]RIKEN AIP
{yaohungt, shaojieb, morency, rsalakhu}@cs.cmu.edu, myamada@i.kyoto-u.ac.jp
https://github.com/yaohungt/TransformerDissection

Abstract

Transformer is a powerful architecture that achieves superior performance on various sequence learning tasks, including neural machine translation, language understanding, and sequence prediction. At the core of the Transformer is the attention mechanism, which concurrently processes all inputs in the streams. In this paper, we present a new formulation of attention via the lens of the kernel. To be more precise, we realize that the attention can be seen as applying kernel smoother over the inputs with the kernel scores being the similarities between inputs. This new formulation gives us a better way to understand individual components of the Transformer's attention, such as the better way to integrate the positional embedding. Another important advantage of our kernel-based formulation is that it paves the way to a larger space of composing Transformer's attention. As an example, we propose a new variant of Transformer's attention which models the input as a product of symmetric kernels. This approach achieves competitive performance to the current state of the art model with less computation. In our experiments, we empirically study different kernel construction strategies on two widely used tasks: neural machine translation and sequence prediction.

1 Introduction

Transformer (Vaswani et al., 2017) is a relative new architecture which outperforms traditional deep learning models such as Recurrent Neural Networks (RNNs) (Sutskever et al., 2014) and Temporal Convolutional Networks (TCNs) (Bai et al., 2018) for sequence modeling tasks across neural machine translations (Vaswani et al., 2017), language understanding (Devlin et al., 2018), sequence prediction (Dai et al., 2019), image generation (Child et al., 2019), video activity classification (Wang et al., 2018), music generation (Huang et al., 2018a), and multimodal sentiment analysis (Tsai et al., 2019a). Instead of performing recurrence (e.g., RNN) or convolution (e.g., TCN) over the sequences, Transformer is a feed-forward model that concurrently processes the entire sequence. At the core of the Transformer is its attention mechanism, which is proposed to integrate the dependencies between the inputs. There are up to three types of attention within the full Transformer model as exemplified with neural machine translation application (Vaswani et al., 2017): 1) Encoder self-attention considers the source sentence as input, generating a sequence of encoded representations, where each encoded token has a global dependency with other tokens in the input sequence. 2) Decoder self-attention considers the target sentence (e.g., predicted target sequence for translation) as input, generating a sequence of decoded representations[1], where each decoded token depends on previous decoded tokens. 3) Decoder-encoder attention considers both encoded and decoded sequences, generating a sequence with the same length as the decoded sequence. It should be noted that some applications has only the decoder self-attention such as sequence prediction (Dai et al., 2019). In all cases, the Transformer's attentions follow the same general mechanism.

At the high level, the attention can be seen as a weighted combination of the input sequence, where the weights are determined by the similarities between elements of the input sequence. We note that this operation is order-agnostic to the permutation in the input se-

[1]The generated sequence can be regarded as a translated sequence (i.e., translating from the encoded sequence), where each generated token depends on all tokens in the encoded sequence.

quence (order is encoded with extra positional embedding (Vaswani et al., 2017; Shaw et al., 2018; Dai et al., 2019)). The above observation inspires us to connect Transformer's attention to kernel learning (Scholkopf and Smola, 2001): they both concurrently and order-agnostically process all inputs by calculating the similarity between the inputs. Therefore, in the paper, we present a new formulation for Transformer's attention via the lens of kernel. To be more precise, the new formulation can be interpreted as a kernel smoother (Wasserman, 2006) over the inputs in a sequence, where the kernel measures how similar two different inputs are. The main advantage of connecting attention to kernel is that it opens up a new family of attention mechanisms that can relate to the well-established literature in kernel learning (Scholkopf and Smola, 2001). As a result, we develop a new variant of attention which simply considers a product of symmetric kernels when modeling non-positional and positional embedding.

Furthermore, our proposed formulation highlights naturally the main components of Transformer's attention, enabling a better understanding of this mechanism: recent variants of Transformers (Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019; Lee et al., 2018; Wang et al., 2018; Tsai et al., 2019a) can be expressed through these individual components. Among all the components, we argue that the most important one is the construction of the kernel function. We empirically study multiple kernel forms and the ways to integrate positional embedding in neural machine translation (NMT) using IWSLT'14 German-English (De-En) dataset (Edunov et al., 2017) and sequence prediction (SP) using WikiText-103 dataset (Merity et al., 2016).

2 Attention

This section aims at providing an understanding of attention in Transformer via the lens of kernel. The inspiration for connecting the kernel (Scholkopf and Smola, 2001) and attention instantiates from the observation: both operations concurrently processes all inputs and calculate the similarity between the inputs. We first introduce the background (i.e., the original formulation) of attention and then provide a new reformulation within the class of kernel smoothers (Wasserman, 2006). Next, we show that this new formulation allows us to explore new family of attention while at the same time offering a framework to categorize previous attention variants (Vaswani et al., 2017; Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019; Lee et al., 2018; Wang et al., 2018; Tsai et al., 2019a). Last, we present a new form of attention, which requires fewer parameters and empirically reaches competitive performance as the state-of-the-art models.

For notation, we use lowercase representing a vector (e.g., x), bold lowercase representing a matrix (e.g., $\mathbf{x}$), calligraphy letter denoting a space (e.g., $\mathcal{X}$), and S denoting a set. To relate the notations in sequence to sequence learning (Vaswani et al., 2017), x represents a specific element of a sequence, $\mathbf{x} = [x_1, x_2, \cdots, x_T]$ denotes a sequence of features, $S_{\mathbf{x}} = \{x_1, x_2, \cdots, x_T\}$ represents the set with its elements being the features in sequence $\mathbf{x}$, and we refer the space of set $S_{\mathbf{x}}$ as $\mathcal{S}$.

2.1 Technical Background

Unlike recurrent computation (Sutskever et al., 2014) (i.e., RNNs) and temporal convolutional computation (Bai et al., 2018) (i.e., TCNs), Transformer's attention is an *order-agnostic* operation given the order in the inputs (Vaswani et al., 2017). Hence, in the presentation of the paper, we consider the inputs as a set instead of a sequence. When viewing sequence as a set, we lose the temporal (positional) information in inputs which is often crucial for sequence modeling (Sutskever et al., 2014). As a result, Transformer (Vaswani et al., 2017) introduced positional embedding to indicate the positional relation for the inputs. Formally, a sequence $\mathbf{x} = [x_1, x_2, \cdots, x_T]$ defines each element as $x_i = (f_i, t_i)$ with $f_i \in \mathcal{F}$ being the non-temporal feature at time i and $t_i \in \mathcal{T}$ as an temporal feature (or we called it positional embedding). Note that f_i can be the word representation (in neural machine translation (Vaswani et al., 2017)), a frame in a video (in video activity recognition (Wang et al., 2018)), or a music unit (in music generation (Huang et al., 2018b)). t_i can be a mixture of sine and cosine functions (Vaswani et al., 2017) or parameters that can be learned during back-propagation (Dai et al., 2019; Ott et al., 2019). The feature vector are defined over a joint space $\mathcal{X} := (\mathcal{F} \times \mathcal{T})$. The resulting permutation-

invariant set is: $S_{\mathbf{x}} = \{x_1, x_2, \cdots, x_T\} = \{(f_1, t_1), (f_2, t_2), \cdots, (f_T, t_T)\}$.

Followed the definition by Vaswani et al. (2017), we use queries(q)/keys(k)/values(v) to represent the inputs for the attention. To be more precise, $x_{\{q/k/v\}}$ is used for denoting a query/key/value data in the query/key/value sequence $\mathbf{x}_{\{q/k/v\}}$ ($x_{\{q/k/v\}} \in S_{\mathbf{x}_{\{q/k/v\}}}$) with $S_{\mathbf{x}_{\{q/k/v\}}}$ being its set representation. We note that the input sequences are the same ($\mathbf{x}_q = \mathbf{x}_k$) for self-attention and are different ($\mathbf{x}_q$ from decoder and $\mathbf{x}_k$ from encoder) for encoder-decoder attention.

Given the introduced notation, the attention mechanism in original Transformer (Vaswani et al., 2017) can be presented as:

$$\begin{aligned}&\text{Attention}(x_q\ ;\ S_{\mathbf{x}_k}) \\ &= \text{softmax}\left(\frac{x_q W_q (\mathbf{x}_k W_k)^\top}{\sqrt{d_k}}\right)\mathbf{x}_k W_v\end{aligned} \tag{1}$$

with $x_q = f_q + t_q$, $\mathbf{x}_k = \mathbf{f}_k + \mathbf{t}_k$, $W_{q/k/v}$ being the weight, and d_k being the feature dimension of $\mathbf{x}_k W_k$. Decoder self-attention further introduces a mask to block the visibility of elements in $S_{\mathbf{x}_k}$ to x_q. Particularly, decoder self-attention considers the decoded sequence as inputs ($\mathbf{x}_k = \mathbf{x}_q$), where the decoded token at time t is not allowed to access the future decoded tokens (i.e., tokens decoded at time greater than t). On the contrary, encoder self-attention and decoder-encoder attention consider no additional mask to Eq. (1).

Recent work (Shaw et al., 2018; Dai et al., 2019; Huang et al., 2018b; Child et al., 2019; Lee et al., 2018; Parmar et al., 2018; Tsai et al., 2019a) proposed modifications to the Transformer for the purpose of better modeling inputs positional relation (Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019), appending additional keys in $S_{\mathbf{x}_k}$ (Dai et al., 2019), modifying the mask applied to Eq. (1) (Child et al., 2019), or applying to distinct feature types (Lee et al., 2018; Parmar et al., 2018; Tsai et al., 2019a). These works adopt different designs of attention as comparing to the original form (Eq. (1)). In our paper, we aim at providing an unified view via the lens of kernel.

2.2 Reformulation via the Lens of Kernel

We now provide the intuition to reformulate Eq. (1) via the lens of kernel. First, the softmax function can be realized as a probability function for x_q observing the keys $\{x_k\}$s in $S_{\mathbf{x}_k}$ ($S_{\mathbf{x}_k}$ is the set representation of sequence $\mathbf{x}_k$). The probability is determined by the dot product between x_q and x_k with additional mappings W_q/W_k and scaling by d_k, which we note the dot-product operation is an instance of kernel function. We also introduce a set filtering function $M(x_q, S_{\mathbf{x}_k}) : \mathcal{X} \times \mathcal{S} \to \mathcal{S}$ which returns a set with its elements that operate with (or are connected/visible to) x_q. The filtering function $M(\cdot,\cdot)$ plays as the role of the mask in decoder self-attention (Vaswani et al., 2017). Putting these altogether, we re-represent Eq. (1) into the following definition.

Definition 1. *Given a non-negative kernel function $k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, a set filtering function $M(\cdot,\cdot) : \mathcal{X} \times \mathcal{S} \to \mathcal{S}$, and a value function $v(\cdot) : \mathcal{X} \to \mathcal{Y}$, the Attention function taking the input of a query feature $x_q \in \mathcal{X}$ is defined as*

$$\begin{aligned}&\text{Attention}\Big(x_q\ ;\ M(x_q, S_{\mathbf{x}_k})\Big) \\ &= \sum_{x_k \in M(x_q, S_{\mathbf{x}_k})} \frac{k(x_q, x_k)}{\sum_{x_k' \in M(x_q, S_{\mathbf{x}_k})} k(x_q, x_k')} v(x_k).\end{aligned} \tag{2}$$

The Definition 1 is a class of linear smoothers (Wasserman, 2006) with kernel smoothing:

$$\begin{aligned}&\sum_{x_k \in M(x_q, S_{\mathbf{x}_k})} \frac{k(x_q, x_k)}{\sum_{x_k' \in M(x_q, S_{\mathbf{x}_k})} k(x_q, x_k')} v(x_k) \\ &= \mathbb{E}_{p(x_k|x_q)}\big[v(x_k)\big],\end{aligned}$$

where $v(x_k)$ outputs the "values" and $p(x_k|x_q) = \frac{k(x_q,x_k)}{\sum_{x_k' \in M(x_q, S_{\mathbf{x}_k})} k(x_q,x_k')}$ is a probability function depends on k and N when $k(\cdot,\cdot)$ is always positive. In the prior work (Vaswani et al., 2017), $k(x_q, x_k) = \exp\left(\langle x_q W_q, x_k W_k\rangle / \sqrt{d_k}\right)$ and $v(x_k) = x_k W_v$. Note that the kernel form $k(x_q, x_k)$ in the original Transformer (Vaswani et al., 2017) is a asymmetric exponential kernel with additional mapping W_q and W_k (Wilson et al., 2016; Li et al., 2017)[2].

The new formulation defines a larger space for composing attention by manipulating its individual components, and at the same time it is

[2]We note that rigorous definition of kernel function (Scholkopf and Smola, 2001) requires the kernel to be semi-positive definite and symmetric. While in the paper, the discussion on kernel allows it to be non-semi-positive definite and asymmetric. In Section 3, we will examine the kernels which are semi-positive and symmetric.

able to categorize different variants of attention in prior work (Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019; Lee et al., 2018; Wang et al., 2018; Tsai et al., 2019a). In the following, we study these components by dissecting Eq. (2) into: 1) kernel feature space $\mathcal{X}$, 2) kernel construction $k(\cdot,\cdot)$, 3) value function $v(\cdot)$, and 4) set filtering function $M(\cdot,\cdot)$.

2.2.1 Kernel Feature Space $\mathcal{X}$

In Eq. (2), to construct a kernel on $\mathcal{X}$, the first thing is to identify the kernel feature space $\mathcal{X}$. In addition to modeling sequences like word sentences (Vaswani et al., 2017) or music signals (Huang et al., 2018b), the Transformer can also be applied to images (Parmar et al., 2018), sets (Lee et al., 2018), and multimodal sequences (Tsai et al., 2019a). Due to distinct data types, these applications admit various kernel feature space:

(i) Sequence Transformer (Vaswani et al., 2017; Dai et al., 2019):

$$\mathcal{X} := (\mathcal{F} \times \mathcal{T})$$

with $\mathcal{F}$ being non-positional feature space and $\mathcal{T}$ being the positional embedding space of the position in the sequence.

(ii) Image Transformer (Parmar et al., 2018):

$$\mathcal{X} := (\mathcal{F} \times \mathcal{H} \times \mathcal{W})$$

with $\mathcal{F}$ being non-positional feature space, $\mathcal{H}$ being the positional space of the height in an image, and $\mathcal{W}$ being the positional space of the width in an image.

(iii) Set Transformer (Lee et al., 2018) and Non-Local Neural Networks (Wang et al., 2018):

$$\mathcal{X} := (\mathcal{F})$$

with no any positional information present.

(iv) Multimodal Transformer (Tsai et al., 2019a):

$$\mathcal{X} := (\mathcal{F}^{\ell} \times \mathcal{F}^{v} \times \mathcal{F}^{a} \times \mathcal{T})$$

with $\mathcal{F}^{\ell}$ representing the language feature space, $\mathcal{F}^{v}$ representing the vision feature space, $\mathcal{F}^{a}$ representing the audio feature space, and $\mathcal{T}$ representing the temporal indicator space.

For the rest of the paper, we will focus on the setting for sequence Transformer $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$ and discuss the kernel construction on it.

2.2.2 Kernel Construction and the Role of Positional Embedding $k(\cdot,\cdot)$

The kernel construction on $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$ has distinct design in variants of Transformers (Vaswani et al., 2017; Dai et al., 2019; Huang et al., 2018b; Shaw et al., 2018; Child et al., 2019). Since now the kernel feature space considers a joint space, we will first discuss the kernel construction on $\mathcal{F}$ (the non-positional feature space) and then discuss how different variants integrate the positional embedding (with the positional feature space $\mathcal{T}$) into the kernel.

Kernel construction on $\mathcal{F}$. All the work considered the scaled asymmetric exponential kernel with the mapping W_q and W_k (Wilson et al., 2016; Li et al., 2017) for non-positional features f_q and f_k:

$$k_{\exp}(f_q, f_k) = \exp\left(\frac{\langle f_q W_q, f_k W_k\rangle}{\sqrt{d_k}}\right). \quad (3)$$

Note that the usage of asymmetric kernel is also commonly used in various machine learning tasks (Yilmaz, 2007; Tsuda, 1999; Kulis et al., 2011), where they observed the kernel form can be flexible and even non-valid (i.e., a kernel that is not symmetric and positive semi-definite). In Section 3, we show that symmetric design of the kernel has similar performance for various sequence learning tasks, and we also examine different kernel choices (i.e., linear, polynomial, and rbf kernel).

Kernel construction on $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$. The designs for integrating the positional embedding t_q and t_k are listed in the following.

(i) Absolute Positional Embedding (Vaswani et al., 2017; Dai et al., 2019; Ott et al., 2019): For the original Transformer (Vaswani et al., 2017), each t_i is represented by a vector with each dimension being sine or cosine functions. For learned positional embedding (Dai et al., 2019; Ott et al., 2019), each t_i is a learned parameter and is fixed for the same position for different sequences. These works defines the feature space as the direct sum of its temporal and non-temporal space: $\mathcal{X} = \mathcal{F} \oplus \mathcal{T}$. Via the lens of kernel, the kernel similarity is defined as

$$k\Big(x_q, x_k\Big) := k_{\exp}\Big(f_q + t_q, f_k + t_k\Big). \quad (4)$$

(ii) Relative Positional Embedding in Transformer-XL (Dai et al., 2019): t represents the indicator of

the position in the sequence, and the kernel is chosen to be asymmetric of mixing sine and cosine functions:

$$k\Big(x_q, x_k\Big) := k_{\exp}\Big(f_q, f_k\Big) \cdot k_{f_q}\Big(t_q, t_k\Big) \quad (5)$$

with $k_{f_q}\Big(t_q, t_k\Big)$ being an asymmetric kernel with coefficients inferred by f_q: $\log k_{f_q}\Big(t_q, t_k\Big) = \sum_{p=0}^{\lfloor d_k/2 \rfloor - 1} c_{2p} \sin(\frac{t_q - t_k}{10000^{\frac{2p}{512}}}) + c_{2p+1} \cos(\frac{t_q - t_k}{10000^{\frac{2p}{512}}})$ with $[c_0, \cdots, c_{d_k-1}] = f_q W_q W_R$ where W_R is an learned weight matrix. We refer readers to Dai et al. (2019) for more details.

(iii) Relative Positional Embedding of Shaw et al. (2018) and Music Transformer (Huang et al., 2018b): $t.$ represents the indicator of the position in the sequence, and the kernel is modified to be indexed by a look-up table:

$$k\Big(x_q, x_k\Big) := L_{t_q - t_k, f_q} \cdot k_{\exp}\Big(f_q, f_k\Big), \quad (6)$$

where $L_{t_q - t_k, f_q} = \exp(f_q W_q a_{t_q - t_k})$ with $a.$ being a learnable matrix having matrix width to be the length of the sequence. We refer readers to Shaw et al. (2018) for more details.

Dai et al. (2019) showed that the way to integrate positional embedding is better through Eq. (5) than through Eq. (6) and is better through Eq. (6) than through Eq. (4). We argue the reason is that if viewing f_i and t_i as two distinct spaces $\Big(\mathcal{X} := (\mathcal{F} \times \mathcal{T})\Big)$, the direct sum $x_i = f_i + t_i$ may not be optimal when considering the kernel score between x_q and x_k. In contrast, Eq. (5) represents the kernel as a product of two kernels (one for f_i and another for t_i), which is able to capture the similarities for both temporal and non-temporal components.

2.2.3 Value Function $v(\cdot)$

The current Transformers consider two different value function construction:

(i) Original Transformer (Vaswani et al., 2017) and Sparse Transformer (Child et al., 2019):

$$v(x_k) = v((f_k, t_k)) := (f_k + t_k) W_v. \quad (7)$$

(ii) Transformer-XL (Dai et al., 2019), Music Transformer (Huang et al., 2018b), Self-Attention with Relative Positional Embedding (Shaw et al., 2018):

$$v(x_k) = v((f_k, t_k)) := f_k W_v. \quad (8)$$

Compared Eq. (7) to Eq. (8), Eq. (7) takes the positional embedding into account for constructing the value function. In Section 3, we empirically observe that constructing value function with Eq. (8) constantly outperforms the construction with Eq. (7), which suggests that we do not need positional embedding for value function.

2.2.4 Set Filtering Function $M(\cdot, \cdot)$

In Eq. (2), the returned set by the set filtering function $M(x_q, S_{\mathbf{x}_k})$ defines how many keys and which keys are operating with x_q. In the following, we itemize the corresponding designs for the variants in Transformers:

(i) Encoder Self-Attention in original Transformer (Vaswani et al., 2017): For each query x_q in the encoded sequence, $M(x_q, S_{\mathbf{x}_k}) = S_{\mathbf{x}_k}$ contains the keys being all the tokens in the encoded sequence. Note that encoder self-attention considers $\mathbf{x}_q = \mathbf{x}_k$ with $\mathbf{x}_q$ being the encoded sequence.

(ii) Encoder-Decoder Attention in original Transformer (Vaswani et al., 2017): For each query x_q in decoded sequence, $M(x_q, S_{\mathbf{x}_k}) = S_{\mathbf{x}_k}$ contains the keys being all the tokens in the encoded sequence. Note that encode-decoder attention considers $\mathbf{x}_q \neq \mathbf{x}_k$ with $\mathbf{x}_q$ being the decoded sequence and $\mathbf{x}_k$ being the encoded sequence.

(iii) Decoder Self-Attention in original Transformer (Vaswani et al., 2017): For each query x_q in the decoded sequence, $M(x_q, S_{\mathbf{x}_k})$ returns a subset of $S_{\mathbf{x}_k}$ ($M(x_q, S_{\mathbf{x}_k}) \subset S_{\mathbf{x}_k}$). Note that decoder self-attention considers $\mathbf{x}_q = \mathbf{x}_k$ with $\mathbf{x}_q$ being the decoded sequence. Since the decoded sequence is the output for previous timestep, the query at position i can only observe the keys being the tokens that are decoded with position $< i$. For convenience, let us define S_1 as the set returned by original Transformer (Vaswani et al., 2017) from $M(x_q, S_{\mathbf{x}_k})$, which we will use it later.

(iv) Decoder Self-Attention in Transformer-XL (Dai et al., 2019): For each query x_q in the decoded sequence, $M(x_q, S_{\mathbf{x}_k})$ returns a set containing S_1 and additional memories ($M(x_q, S_{\mathbf{x}_k}) = S_1 + S_{mem}, M(x_q, S_{\mathbf{x}_k}) \supset S_1$). S_{mem} refers to additional memories.

(v) Decoder Self-Attention in Sparse Transformer (Child et al., 2019): For each query x_q in the decoded sentence, $M(x_q, S_{\mathbf{x}_k})$ returns a subset of S_1 ($M(x_q, S_{\mathbf{x}_k}) \subset S_1$).

To compare the differences for various designs, we see the computation time is inversely propor-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.