2022

KSAT: Knowledge-infused Self Attention Transformer - Integrating Multiple Domain-Specific Contexts

Kaushik Roy
Yuxin Zi
Vignesh Narayanan
Manas Gaur
Amit P. Sheth

Follow this and additional works at: https://scholarcommons.sc.edu/aii_fac_pub

Part of the Computer Engineering Commons, and the Electrical and Computer Engineering Commons
KSAT: Knowledge-infused Self Attention Transformer
- Integrating Multiple Domain-Specific Contexts

Kaushik Roy  
Artificial Intelligence Institute South Carolina  
University of South Carolina  
kaushikr@email.sc.edu

Yuxin Zi  
Artificial Intelligence Institute South Carolina  
University of South Carolina  
yzi@email.sc.edu

Vignesh Narayanan  
Artificial Intelligence Institute South Carolina  
University of South Carolina  
vignar@sc.edu

Manas Gaur  
Dept of Computer Science and Engineering  
University of Maryland, Baltimore County  
manas@umbc.edu

Amit Sheth  
Artificial Intelligence Institute South Carolina  
University of South Carolina  
amit@sc.edu

Abstract

Domain-specific language understanding requires integrating multiple pieces of relevant contextual information. For example, we see both suicide and depression-related behavior (multiple contexts) in the text “I have a gun and feel pretty bad about my life, and it wouldn’t be the worst thing if I didn’t wake up tomorrow”. Domain specificity in self-attention architectures is handled by fine-tuning on excerpts from relevant domain specific resources (datasets and external knowledge - medical textbook chapters on mental health diagnosis related to suicide and depression). We propose a modified self-attention architecture Knowledge-infused Self Attention Transformer (KSAT) that achieves the integration of multiple domain-specific contexts through the use of external knowledge sources. KSAT introduces knowledge-guided biases in dedicated self-attention layers for each knowledge source to accomplish this. In addition, KSAT provides mechanics for controlling the trade-off between learning from data and learning from knowledge. Our quantitative and qualitative evaluations show that (1) the KSAT architecture provides novel human-understandable ways to precisely measure and visualize the contributions of the infused domain contexts, and (2) KSAT performs competitively with other knowledge-infused baselines and significantly outperforms baselines that use fine-tuning for domain-specific tasks.

1 Motivation

Solving domain-specific tasks such as mental health diagnosis (MHD), and triaging, requires integrating relevant contextual information from data and knowledge sources. Self-Attention based Language Models (SAMs) capture an aggregated broader context from domain-agnostic, voluminous training corpora [1]. Fine-tuning SAMs on domain-specific corpora achieves domain-specific context capture [2, 3]. However, SAM architectures are black-box in nature [4]. Consequently, fine-tuned
SAM architectures do not lend themselves to the robust evaluation of the open research aims: (R1) Relevant domain-specific context coverage, and (R2) The influence of knowledge context traded-off against the data context in downstream tasks [5, 6]. We propose a modified self-attention architecture Knowledge-infused Self Attention Transformer (KSAT) to address these aims. KSAT performs well on select domain-specific tasks (see Task Description, Data, and External Knowledge Sources) while lending itself to a robust human-understandable evaluation of R1 and R2. Thus KSAT provides a substantial step towards fostering AI-user trust, and satisfaction [7, 8].

2 Background

2.1 Related Work

Prior approaches that are relevant to R1 and R2 and incorporate multiple knowledge contexts can be broadly categorized based on the knowledge-infusion technique as (1) knowledge modulated SAMs and (2) knowledge infused input embedding-based SAMs [9, 10]. The former uses knowledge to guide the self-attention mechanism in SAMs, and the latter embeds the knowledge into a vector space before passing the inputs into SAMs. Here, we briefly summarize their contributions towards R1 and R2. Both Category (1), and Category (2) methods’ domain coverage is evaluated through performance on domain-specific task descriptions (R1). These methods’ ablations highlight contributions of knowledge context (R2). However, inspecting the numerical outputs from the model components (projection matrices and vectors) does not easily lend themselves to human-understandable scrutiny. Explainable AI techniques (post-processing of the numerical outputs that transform them into human-understandable information) are required to confirm the author(s) perspectives [11]. Post-processing-based explanations are local approximations of the SAM reasoning for particular inputs and therefore do not present the global picture, casting doubts on the SAM evaluation validity. KSAT presents a SAM architecture whose numerical outputs lend themselves to robust human-understandable evaluations of R1 and R2.

2.2 Task Description, Data, and External Knowledge Sources

Although the KSAT architecture broadly applies to any domain-specific task, we choose the specific task of Mental Health Diagnostic Assistance for Suicidal Tendencies by Gaur et al. [12]. We denote this dataset as MHDA. The data contains high-quality expert annotations on Reddit posts from suicide-related subreddits. The annotation method ensures minimal noise from measurement artifacts and high agreement among the expert annotators. We use the clinically established diagnostic process information contained in the Columbia Suicide Severity Rating Scale (CSSRS) for knowledge contexts. Figure 1 (a, b) illustrate the various contexts (each tree path represents a context) under which suicidal patterns can arise. The task is to predict the suicidal patterns, namely - indication, ideation1, ideation2, and behavior or attempt. Figure 1 (c) shows examples from the MHDA dataset, augmented with knowledge context annotations. We denote the augmented dataset as k-MHDA. k-MHDA contains knowledge context annotations at the post and sentence level (see Figure 1 (c)). We defer construction details of k-MHDA from the CSSRS knowledge and the MHDA data to the appendix Section Constructing k-MHDA as it is not the main focus of the paper\(^1\).

3 KSAT - Proposed Architecture

For an input $x$ in the vanilla SAM layer $l$, a $CLS$ token is introduced that encodes the information in the data for downstream tasks. For example, for classification, $Z_{CLS}$, the representation of the $CLS$ token from the final layer $L$ is passed through a softmax layer which outputs class probabilities.

In the KSAT layer, we introduce a $Z_{KCLS}$ token that encodes the knowledge for that layer. The $Z_{KCLS}$ values are determined by the graph context between the input $x$ sentences and the concepts

\(^1\)We will release the k-MHDA dataset, along with code to construct it along with the KSAT code for reproducibility of results.
in the knowledge context annotations for input $x$. As an example, consider the inputs $x$ to be like
the posts **Post X1** and **Post X2** shown in Figure 1(c), then the graphs in Figure 1 (d), illustrate the
graph contexts that show sentence-concept connections. One would expect that the $Z_{KCLS}$ encoding
for the sentences S1 and S2 in both posts would be similar as they have similar graph contexts. The
$\text{KG}_{\text{bias}}$ term ensures that the values in $Z_{KCLS}$ captures this behavior (see Figure 2 (b)).

### 3.1 The KSAT layer

In contrast with the vanilla SAM layer, every KSAT layer has the following key differences:

#### 3.1.1 Layer Parameters and Outputs

Let $Y$ denote the set of suicidal outcomes pertaining to each context: \{Indication or None, Ideation1, Ideation2, Behavior or Attempt\} (see Figure 1 (a)). Let $x$ denote an input post (see Figure 1 (c) for example posts). For each input $x$, every KSAT layer $l$ outputs a vector of probabilities for every
outcome $y \in Y$ given by Equation 1:

$$P_l^{\text{ksat}}(y|x) = \sigma(W^T(\alpha_l Z_{\text{KCLS}}(x) + (1 - \alpha_l)Z_{\text{CLS}}(x)) + \text{\text{KG}_{\text{bias}}}(x))$$

(1)

$W$, $Z_{\text{CLS}}(x)$, and $Z_{\text{KCLS}}(x)$ are of dimension size $384 \times 1$ as we use the sentence-transformer
published by Reimers et al. fine-tuned on the MHDA corpus, for embedding inputs [13]. Recall that
the term $\text{KG}_{\text{bias}}$ is used to ensure that $Z_{\text{CLS}}(x)$ encodes the knowledge context represented as a
graph (details in Encoding Graph Contexts in Layers). The dimension size of $\text{KG}_{\text{bias}}$ is $1 \times 1$. The
scalar term $\alpha_l$ is used as a trade-off factor modulating the contributions between data and knowledge
contexts using a convex combination (see Equation 1). We contrast KSAT layer outputs against the
vanilla SAM layer, where a similar probability vector is output only in the final layer and not at
every layer (see Figure 2). The parameters within the vanilla SAM layer (the query, key, and value
projection matrices) used to compute the self-attention matrix are retained in the KSAT layer [1].
Unlike in the vanilla SAM layer, where the parameters may or may not be shared across layers, the
parameters are not shared across different KSAT layers as different layers encode different knowledge contexts.

3.1.2 Encoding Graph Contexts in Layers

Even though every KSAT layer \( l \) outputs a vector of probabilities for every outcome \( y \in \{\text{Indication or None}, \text{Ideation1}, \text{Ideation2}, \text{Behavior or Attempt}\} \), representing each context, it encodes only a single context in \( Z_{KCLS} \). We show how to compute the \( KG_{bias} \) term by referencing the posts \( \text{Post X1} \) and \( \text{Post X2} \) and the graph contexts \text{Ideation1} \) and \text{Behavior or Attempt} \) from Figure 1 (c,d).

Computing Sentence-Concept Connection Vectors: We first compute a sentence-concept connection vector for each sentence in the posts. For \( \text{Post X1} \), both sentence \( S1 \) and \( S2 \) are connected to the concept \( C1 \) in the concept set \{\( C1,C2,C3 \}\) (see Figure 1 (b)). Therefore both their sentence-concept connection vectors are computed as: \([1, 0, 0]\). Similarly for \( \text{Post X2} \), the sentence-concept connection vectors for both \( S1 \) and \( S2 \) are computed as: \([1, 1, 1]\).

Computing Graph Context Distances: Recall that KSAT layers take as input a single post \( (x \text{ in Equation 1}) \). Denoting the sentence-concept vectors for a sentence \( S1 \) as \( C_{S1} \), to compute \( KG_{bias}(x) \), we first need to compute graph context distances for the posts \( \text{Post X1} \) and \( \text{Post X2} \): \( d(C_{S1}, C_{S2}) \). If the posts had more than two sentences the graph context distances would include all pairs (Eg: \( d(C_{S1}, C_{S2}), d(C_{S2}, C_{S3}), d(C_{S1}, C_{S3}) \) for three sentences). The term \( d(C_{S1}, C_{Sj}) \) for a pair of sentences \( (S_i, S_j) \) in post \( x \) captures the graph context-based distance between the sentences. Intuitively, sentences that have equivalent graph contexts should have \( d(S_i, S_j) = 0 \). We use hamming distance in our experiments.

The \( KG_{bias} \) term for an input \( x \) is thus given by Equation 2 as:

\[
KG_{bias}(x) = - \sum_{(S_i, S_j) \in x} \frac{(Z_{KCLS}(x)[S_i] - Z_{KCLS}(x)[S_j])^2}{d(S_i, S_j) + \epsilon} \tag{2}
\]

The formulation for the \( KG_{bias}(x) \) term in Equation 2 encourages the \( Z_{KCLS} \) representations of sentences that are in the same graph context to be similar. The \( \epsilon \) term is to prevent dividing by zero errors.

3.2 Aggregating KSAT Layer Outputs

Combining KSAT layer probabilities given by Equation 1 is application domain dependent. For the suicidal outcomes in the set: \{\text{Indication or None}, \text{Ideation1}, \text{Ideation2}, \text{Behavior or Attempt}\}, it is reasonable to expect that suicidal ideation (both \text{Ideation1} and \text{Ideation2}) precedes the act of attempting suicide (\text{Behavior or Attempt}). In other application domains, the contexts could be independent of each other.

In our experiments, we stack four KSAT layers corresponding to the outcomes \text{Indication or None}, \text{Ideation1}, \text{Ideation2}, \text{Behavior or Attempt}, in that order (the order is derived from the tree structure in Figure 1 (a)). Typically, for dependent probability outcomes, \( X \) and \( Y \), where \( X \) precedes \( Y \), the probability \( P(X, Y) \) would be modeled as \( P(Y \mid X)P(X) \). However, since we stack KSAT layers in a particular order, we use a product approximation \( P(X, Y) = P(X)P(Y) \) as information is propagated upwards through the KSAT layers. Thus, the final layer probabilities from the KSAT layers is computed using Equation 3 as:

\[
P_{final}^{ksat}(y|x) = \prod_{l} P_{l}^{ksat}(y|x), \tag{3}
\]

where \( P_{l}^{ksat}(y|x) \) is given by Equation 1
4 KSAT - Results and Analysis

Recall the research aims that KSAT addresses when integrating multiple contexts - (R1) Relevant domain-specific context coverage, and (R2) The influence of knowledge context traded-off against the data context in downstream tasks (see Section Motivation).

4.1 KSAT - Quantitative Results (Addresses R1)

Table 1 shows the accuracy / AUC-ROC scores (rounded-off) for KSAT vs two best-performing fine-tuned transformer models, and knowledge infused baseline models K-type(i) and K-type(ii) (see section Related Work). Due to space concerns we describe details of the models K-type(i) and K-type(ii) in the Appendix Section Construction of the K-type(i) and K-type(ii) baseline models. KSAT outperforms the fine-tuned transformer models and performs comparably with models K-type(i) and K-type(ii) on the k-MHDA dataset (see section Constructing k-MHDA).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>KSAT</th>
<th>K-type(i)</th>
<th>K-type(ii)</th>
<th>XLNET</th>
<th>RoBERTa</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-MHDA</td>
<td>83% / 78</td>
<td>84% / 71%</td>
<td>84% / 72%</td>
<td>68% / 57%</td>
<td>68% / 63%</td>
</tr>
</tbody>
</table>

Table 1: Shows the accuracy / AUC-ROC scores (rounded-off) for KSAT vs fine-tuned transformer models, and knowledge infused baseline models K-type(i) and K-type(ii). KSAT outperforms all fine-tuned transformer models and performs comparably with models K-type(i) and K-type(ii) on the k-MHDA dataset.

4.2 KSAT-Qualitative Results (Addresses R2)

Figure 3 illustrates the final KSAT layer representations (the ZKCLS vectors) and αl values of sample test posts to visualize data and knowledge contexts (see section Layer Parameters and Outputs).

5 Conclusion

We proposed KSAT that integrates multiple contexts and data and shows its utility in the domain-specific use-case of MHDA. Although we test KSAT on MHDA, KSAT applies to other domain-specific tasks that require contextualization from multiple knowledge sources. The architecture of KSAT allows for precise measurement and visualization of the contributions from the different knowledge contexts and the data, thus addressing the research aims R1 and R2. In future work, we will apply KSAT to other downstream domain-specific knowledge-intensive tasks such as conversational...
question answering that requires drawing context from multiple knowledge sources. We will replicate KSAT’s qualitative evaluation in MHDA for robust human-understandable evaluation on future tasks.

References


We can then evaluate the Bernoulli Loss

\[
\mathcal{L}(x, y, \{\theta_i\}) = P(y | x, \{\theta_i\}) \log(P(y | x, \{\theta_i\})) + (1 - P(y | x, \{\theta_i\})) \log(1 - P(y | x, \{\theta_i\}))
\]

as:

\[
P(y | x, \{\theta_i\}) = \sum_{y \in \text{Outcomes}} p_y \prod_{i=1}^{3} \sum_{x_{\text{sub}} \in x} \left( \cos \cdot \text{sim}(x^{R}_{\text{sub}}, q^{R}_i) \geq \theta_i \right) \geq 0.5
\]

Equation 5 determines the presence or absence of concept \( q_i \) in a post sub-fragment \( x_{\text{sub}} \). First, we compute the cosine similarity between their sentence-transformer representations \( x^{R}_{\text{sub}} \) and \( q^{R}_i \). If the resulting value is \( \geq \theta_i \), we determine that the concept \( q_i \) is present in \( x_{\text{sub}} \), else we determine that the concept \( q_i \) is absent in \( x_{\text{sub}} \).

Equation 4 is the algebraic form of the \( \lor \) operation as we determine that concept \( q_i \) is present in the post \( x \), if any of the post fragments \( x_{\text{sub}} \in x \) show presence of concept \( q_i \).
We use grid-search to find a configuration of parameters \( \{\theta_i\} \) and post sub-fragment \( x_{sub} \) that has the maximum value for \( \prod_{(x,y) \in \text{MHDA}} \mathcal{L}(x, y, \{\theta_i\}) \). We vary each individual \( \theta_i \) in the range \([-1, 1]\) (the range of the cosine function) and \( x_{sub} \) takes values from the set \{1, 2, 3\}.

Inference is carried out as it is in a decision tree classifier with the concept presence or absence at each branch, evaluated using Equation 5.

**Knowledge Context Annotation with outputs from grid-search:** The grid-search yielded outputs \( \{\theta_i\} = \{0.3, 0.5, 0.3\} \), and post sub-fragment size \( |x_{sub}| = 1 \) (one sentence). Therefore the post “I don’t feel like waking up and have a gun. Oh well.” is annotated with the knowledge context: \((\text{Concept 1 (T)}, \text{Concept 2 (T)}, \text{Concept 3 (T)}) = \text{Behavior or Attempt}\) as evaluation of Equation 5 determines absence of \text{Concept 1, Concept 2, and Concept 3} in the post sentence “I don’t feel like waking up and have a gun”. The evaluation uses the grid-search outputs of \( \{\theta_i\} \). The second sentence “Oh well” is not necessary to evaluate as we determine a concept’s presence or absence in the post if any of the post fragments \( x_{sub} \) show the presence of the concept.

**B Construction of the K-type(i) and K-type(ii) baseline models**

For our baseline implementations we adapt the state-of-the-art models for each of the model types K-type(i) and K-type(ii) in Section 2.1 for our task description (see Section 2.2. Specifically, we use the model KnowBERT and K-Adapter as K-type(i) and K-type(ii) model instances respectively [9, 10]. For knowledge graph embeddings in the KnowBERT case, we utilize TransE embeddings. Similar to their work, we use the RoBERTa model for the task-specific adapter module. We retain all hyperparameters from the original implementations [14, 15].