SEEN: Structured Event Enhancement Network for Explainable Need Detection of Information Recall Assistance

You-En Lin,[1] **An-Zi Yen,**[2] **Hen-Hsen Huang,**[3] **Hsin-Hsi Chen**[1]
[1] Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
[2] Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan
[3] Institute of Information Science, Academia Sinica, Taiwan
yelin@nlg.csie.ntu.edu.tw, azyen@nycu.edu.tw,
hhhuang@iis.sinica.edu.tw, hhchen@ntu.edu.tw

Abstract

When recalling life experiences, people often forget or confuse life events, which necessitates information recall services. Previous work on information recall focuses on providing such assistance reactively, i.e., by retrieving the life event of a given query. Proactively detecting the need for information recall services is rarely discussed. In this paper, we use a human-annotated life experience retelling dataset to detect the right time to trigger the information recall service. We propose a pilot model–structured event enhancement network (SEEN) that detects life event inconsistency, additional information in life events, and forgotten events. A fusing mechanism is also proposed to incorporate event graphs of stories and enhance the textual representations. To explain the need detection results, SEEN simultaneously provides support evidence by selecting the related nodes from the event graph. Experimental results show that SEEN achieves promising performance in detecting information needs. In addition, the extracted evidence can be served as complementary information to remind users what events they may want to recall.

1 Introduction

People have to deal with many events in their daily life. As time passes, they might forget details about their past experiences. Forgetting the exact name of people or places or things and mixing up life events is a common occurrence. This explains the importance of an information recall system that helps people bring to mind what they are trying to recall. We propose *reactive* and *proactive* service modes for an information recall system (Yen et al., 2021a). In reactive mode, users directly ask the system about their life events, whereas in proactive mode, the system attempts to automatically detect whether users need memory recall assistance and then provides the information they seek to recall. For reactive mode, studies have been done on visual lifelog recall (Gurrin et al., 2016, 2017, 2019, 2020; Chu et al., 2019, 2020), which focuses on the construction of a multimodal retrieval model that enables users to search through photos using textual queries. We propose an information recall system (Yen et al., 2021b) to answer questions about life experiences over a personal knowledge base. In contrast to reactively receiving users' requests, proactive mode, which detects the right time to trigger the information recall service, is still little explored. In this paper, we further propose a pilot study to proactively detect the user's need for information recall assistance.

One common use case of memory recall assistance occurs in human conversation. To identify whether people have difficulties in recalling past experiences, Wang et al. (2018) propose a model to detect speech hesitation. Here, we focus on detecting the need for information recall support in people's narratives. Specifically, we seek to detect the following four situations in narratives to determine whether to trigger the service:

1. If the description of the life event is consistent with the user's past experience, no memory recall assistance is needed.
2. Since people cannot remember every detail of their life experiences, we may unconsciously draw on similar but unrelated events to describe an experience that leads to a conflict with the established facts. It is essential to identify the description that is inconsistent with these facts, and retrieve those facts as an explanation to inform the user.
3. For the case where the narrative ends without relevant events mentioned, the user may have forgotten the events. The system must remind the user of these forgotten events.
4. The user may elaborate on additional events that were not logged before. This additional information could be details about events in lifelogs or they could be previously unlogged events. The system should distinguish

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 5438–5451
December 7-11, 2022

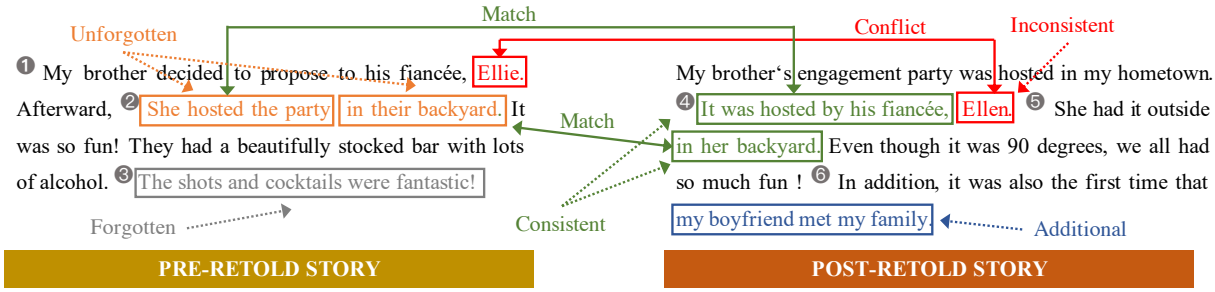

Figure 1: Snippets from two stories in NIR (Left: Pre-Retold, Right: Post-Retold).

whether events are additional or conflict with the facts, and should update the lifelogs with the new information.

To the best of our knowledge, no dataset is available for this purpose. For this reason, we extended the Hippocorpus dataset (Sap et al., 2020) with new life event annotations as cases where users encounter problems and require recall assistance. Sap et al. (2020) invited crowd-workers to write stories about their life experiences, and asked them to write those stories again a few months later. As such, the nature of Hippocorpus meets our requirement. In Hippocorpus, life experiences written the first and the second times are referred to here as *pre-retold* and *post-retold* stories, respectively. The need for information recall is detected by comparing the pre-retold and post-retold stories.

In this paper, we propose a model to identify the event types in post-retold and pre-retold stories. The model is referred to as *structured event enhancement network* (SEEN). A transformer-based language model is used for encoding textual data. To encode the structured information of event description in stories, we construct an event graph by utilizing life event triples. To further capture the relations between events, the results of coreference resolution are incorporated into the event graph. The graph is encoded by the graph attention network (GAT) (Veličković et al., 2018; Brody et al., 2022) and fused with the language model for integrating textual and structured information.

In addition, our model will extract the relevant events in a story pair as support evidence to explain the decision of the prediction. In this way, the user will easily recall the forgotten events. In sum, the contributions of our work are threefold: (1) We introduce the task of detecting the need for information recall in a narrative and providing the related information as the support evidence. (2) We present the NIR dataset,[1] a human-annotated life experience retelling dataset for detecting the **n**eeds of **i**nformation **r**ecall. (3) To detect information needs, we propose the structured event enhancement network (SEEN). The identified event types and extracted support evidence can assist users in recalling their past experiences and clarifying the confusing events.

2 From Hippocorpus to NIR

Sap et al. (2020) constructed Hippocorpus to investigate the difference in the narrative flow between relating life experiences and telling imaginative stories. In this work, we construct NIR by pruning the imaginative stories in Hippocorpus and retaining those stories about real-life events written by crowd-workers at two different times as pre-retold stories and post-retold stories. Following the four situations mentioned in Section 1, we summarize the following five event types from the story pairs in the dataset: *Consistent*, *Inconsistent*, *Additional*, *Forgotten*, and *Unforgotten*. The first three event types occur in the post-retold stories, and the last two event types occur in the pre-retold stories. Figure 1 shows a pair of pre-retold and post-retold stories labeled with these five event types denoted by green, red, blue, gray, and orange boxes, respectively. The numbers in Figure 1 denote the sentences consisting of life events. The details of the five event types are listed as follows:

Consistent: The described event matches the user's life experiences. The event in Sentence (5) is *Consistent* because the event of the brother's fiancée hosting the party in the backyard matches the description in Sentence (2). In this case, the event in Sentence (2) is the support evidence.

Inconsistent: In contrast to *Consistent*, the description is inconsistent with life events. For example, although the description of the fiancée hosting the

[1] https://github.com/ntunlplab/SEEN

party matches Sentence (2), her name in the two stories is different. Thus, the event in Sentence (4) is *Inconsistent*. In other words, if the details of the event description in the post-retold story conflict with the facts described in the pre-retold story, it is an *inconsistent* event.

Additional: This is extra information about a life event that is not previously recorded in the collected lifelogs. The event in Sentence (6) is *Additional* due to the lack of similar event in the pre-retold story.

Forgotten: The life events that have been forgotten, i.e., are not mentioned here. As the event in Sentence (3) does not relate to other events in the post-retold story, it is a *Forgotten* event.

Unforgotten: In contrast to *Forgotten*, the life events in the pre-retold story and also mentioned in the post-retold story belong to *Unforgotten* events. As the events in Sentence (2) are also mentioned in the Sentence (4) and Sentence (5) in the post-retold story, they are *Unforgotten* events.

In our dataset, each life event in the pre-retold and post-retold story stories is labeled with one of five event types. The annotation of relevant events within another story of the story pair is also included to denote as support evidence of the event type. That is, we annotate event types and the corresponding support evidence in the pre-retold and post-retold stories. The construction of the dataset is described in Section 3.

3 Dataset Construction and Analysis

3.1 Life Event Annotation

According to the definition of LiveKB (Yen et al., 2019, 2020) and ConvLogMiner (Kao et al., 2021), we define a life event as a life experience that is related to specific individuals. Note that a sentence may refer to multiple life events. We follow the work of Yen et al. (2019) to extract life events in the triple form *(subject, predicate, object)*. The predicate is also classified into two types: *explicit* and *implicit* to denote whether the predicate is mentioned in the story. The further details are described in Appendix A. Finally, we collected 60,889 events from 2,520 stories consisting of 44,199 sentences. The distribution of explicit and implicit events was 96.9% and 3.1%, respectively.

3.2 Event Type Annotation

Given the life event annotation of each sentence, we invited 11 annotators to label the event types of the life events, where the event types are *Consistent*, *Inconsistent*, *Additional*, *Forgotten*, and *Unforgotten*. Given the pairs of pre-retold and post-retold stories, the annotators were invited to first read the stories to understand the author's experiences. For each story pair, one story is viewed as the reference story, and another story is viewed as the target story. The annotators labeled the event type of each life event in the target story by consulting the reference story. The decision of event type is also based on whether the target story is a pre-retold or post-retold story. In addition, for each story pair, they select the life events in one story that are related to the life events in another story as the support evidence for explaining the event type. Taking Figure 1 as an example, event *(his fiancée Ellen, host, it)* in Sentence (4) is *Inconsistent* since the name of the brother's fiancée conflicts with the event *(my brother, propose to, his fiancée Ellie)* in Sentence (1), although it matches the event *(She, hosted, party)* in Sentence (2). In other words, to identify *Inconsistent* events, comparing the subtle differences in the descriptions of the pre-retold and post-retold stories is essential.

The examination of the annotation quality proceeds similarly to the method mentioned in Appendix A. We randomly sampled 50 story pairs (2,113 events in total) and assigned them to each annotator. An annotator who majored in linguistics was selected as the supervisor. We measured the agreement of each annotator with the supervisor via the F-score. The average event type agreement was a Cohen's kappa score of 0.95. Finally, we collected 1,260 story pairs, with an event-type distribution of *Consistent*, *Inconsistent*, *Additional*, *Forgotten*, and *Unforgotten* events of 11,525, 226, 17,661, 18,773, and 12,704, respectively.

3.3 Event Type Analysis on Age

In general, older people are assumed to need more memory assistance because of the assumption that they are more likely to forget things than younger people. To examine whether elders are indeed more likely to forget or confuse their past experiences, we calculated the average ratio of the five event types in each story pair over eight age groups. In Hippocorpus, 82, 214, 281, 208, 133, 117, 83, and 133 crowd-workers were 18, 25, 30, 35, 40, 45, 50, and 55 years old, respectively. The ratio of each event type in each age group is shown in Figure 2. For better visualization, the bars are presented us-

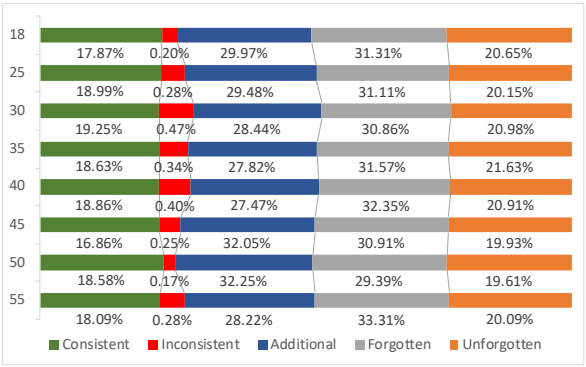

Figure 2: The ratio of each event type in each age group.

ing smoothed and normalized ratios, whereas the numbers under the bars are the average distribution of each event type. The ratio of the *Forgotten* events is similar across all age groups, suggesting that both older people and younger people require information recall support. Hereafter, we view people over or equal to 50-years old as the *50-and-above* group; those younger than 50-years old are the *below-50* group. Comparing the ratio of *Inconsistent* events between the 50-and-above group and below-50 group, those in the latter group were more likely to confuse life events, where the difference was statistically significant (t-test, $p < 0.05$). This suggests that when younger people recall past experiences, they often confuse details. However, when writing post-retold stories, people in the 50-and-above group preferred not to mention events of which they had only vague impressions. The further analyses are described in Appendix B.

4 Task Formulation

Detection of Information Recall Need: To detect the need for information recall, we propose a novel task that is aimed at determining the event type by comparing a pair of pre-retold story $\mathcal{U}$ and post-retold story $\mathcal{V}$. This can be considered a multi-class classification. We regard one story as the reference story D and compare the event triple in another story (i.e., the target story) $D^{'}$ with all sentences in D to identify the event type. Formally, given a pair of $\mathcal{U}$ and $\mathcal{V}$, the task is to identify the life event type $y_i^{D^{'}}$ of the i-th event triple e_i in $D^{'}$, where $y_i \in \{$*Consistent*, *Inconsistent*, *Additional*, *Forgotten*, *Unforgotten*$\}$, and D' denotes $\mathcal{U}$ or $\mathcal{V}$. On the one hand, for the task of identifying *Consistent*, *Inconsistent*, and *Additional* events, $D = \mathcal{U}$ and $D^{'} = \mathcal{V}$. On the other hand, for the task of identifying *Forgotten* and *Unforgotten* events, $D = \mathcal{V}$ and $D^{'} = \mathcal{U}$.

Support Evidence Extraction: To remind the user which event is forgotten or confused in the proactive mode, providing an explanation is beneficial for memory recall. To this end, we also propose an explanation task to extract the events in D that are related to e_i in $D^{'}$ as evidence to explain the decision of event type. The extracted event triple can also help users recall their life experiences.

5 Structured Event Enhancement Network

Although the pre-trained language models have shown great success on various NLP tasks, some works (Xiao et al., 2021; Wang et al., 2020; Tang et al., 2020) also suggest that the structured information can enhance token representations. We construct an event graph based on life event triples. The event graph is incorporated into our model for capturing fine-grained information of life event relations within a document. Inspired by GreaseLM (Zhang et al., 2022), which incorporates the language model with the external knowledge graph, we initialize the node representations by using the language model. Specifically, we extract the hidden states from different encoder layers of the language model as the node representations. A GAT model is employed to propagate the structured information of the event graph. Then the updated node representations are used for enhancing the token representations in the language model by our fusion mechanism. Figure 3 shows an overview of our proposed structured event enhancement network (SEEN). The details are described as follows.

5.1 Event Graph Construction

To construct an event graph G^D, we regard subjects, predicates, and objects of all events in reference story D as the nodes. Since some subjects or objects may refer to other nodes, the nodes which are connected with the coreference links are merged as one node. Here, the coreference links are obtained by utilizing the coreference resolution model (Lee et al., 2018). Then, for each life event triple, we connect the predicate nodes to the subject and object nodes to create G^D. To enhance the connectivity of G^D, we insert a *Super Node* $\mathcal{S}$ into G^D, and connect it to all the other nodes.

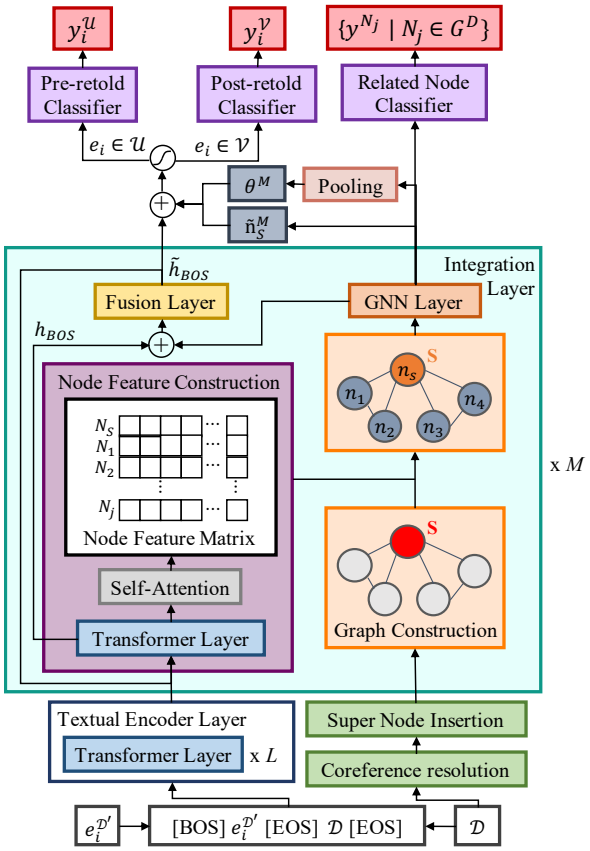

Figure 3: Overview of SEEN.

5.2 Textual Encoder Layer

To encode textual features of reference story D and the i-th event triple e_i in target story D', we concatenate D and $e_i^{D'}$ with the special tokens `[BOS]` and `[EOS]`. The format is `[BOS]` $e_i^{D'}$ `[EOS]` D `[EOS]`. For example, if the goal is to identify the type of i-th life event in $\mathcal{V}$, the input sequence is `[BOS]` $e_i^{\mathcal{V}}$ `[EOS]` $\mathcal{U}$ `[EOS]`, where $e_i^{\mathcal{V}}$ is the concatenation of the components in $e^{\mathcal{V}}$. The output of l-th layer is the hidden states $H^l = \{h^l_{BOS}, h^l_1, ..., h^l_i\}$, where l = 1,..., L. L is a hyperparameter that denotes the number of transformer layers stacked in the textual encoder layer. l=0 is the initial embedding of the tokens.

5.3 Integration Layer

To introduce the structured information of G^D into our language model, we stack M integration layers on the textual encoder layer, where M is a hyperparameter.

Node Feature Construction: Since different layers in the encoder capture different linguistic information for language understanding (Hoover et al., 2020), we initialize the node representations in G^D by using the hidden states of different encoder layers. Each node in G^D is a component in the event triple. Hence, a node can be a text span of the given D. To construct the node feature matrix, we first input H^L into the transformer layer in the integration layer. Then, we construct the initial feature matrix of all nodes in D. Specifically, we extract the hidden states of the `[BOS]` token and the tokens belong to each node. For example, the feature of the j-th node in the m-th integration layer is $[h^m_{BOS}; \|_{t\in T_j} h^m_t]$, where T_j is the token set of the j-th node. Afterward, we concatenate the initial features of each node as the initial feature matrix, and fed the matrix into a self-attention layer. Finally, we take the hidden state of the `[BOS]` token from the self-attention layer's output as the feature of j-th node, which is denote as n^m_j.

$$[n^m_j; \dots] = \text{Attn}([h^m_{BOS}; \Big\|_{t\in T_j} h^m_t]) \cdot W \quad (1)$$

Graph Encoder: After initializing the node features, we exploit the GAT layer to encode the event graph. To learn the representation $\hat{n}^m_j$ of the j-th node N_j from the m-th GAT layer, N_j receives the messages from its neighbor nodes $\mathcal{R}_j$ and computed its feature as Equation 2, where $\alpha^m_{j,j}$ and $\alpha^m_{j,r}$ denote the weights of the j-th node and the r-th neighbor node in m-th GAT layer, respectively. And the attention weight is computed by Equation 3, where $\alpha^m_{s,d}$ denotes the attention weight of the message between the s-th node and the d-th node. The score $x_{s,d}$ is computed by Equation 4. The encoded graph is denoted as $\hat{G}^{D,m} = \{\hat{n}^m_1, ..., \hat{n}^m_j\}$, where $m = 1, ..., M$.

$$\hat{n}^m_j = \alpha^m_{j,j} n^m_j + \sum_{r\in\mathcal{R}_j} \alpha^m_{j,r} n^m_r \quad (2)$$

$$\alpha^m_{s,d} = \frac{x^m_{s,d}}{\sum_{k\in\mathcal{N}_s \bigcup \{s\}} x^m_{s,k}} \quad (3)$$

$$x^m_{a,b} = \exp(W_\tau \text{LeakyRelu}(W_\kappa \cdot [n^m_a; n^m_b]) \quad (4)$$

Fusion Layer: To enhance the language model with the structured information from G^D, we fuse the hidden state of `[BOS]` token $h^m_{BOS} \in \mathbb{R}^\epsilon$ of the m-th transformer layer and the feature of *Super Node* $\hat{n}^m_{\mathcal{S}} \in \mathbb{R}^\delta$ in $\hat{G}^{D,m}$, where ϵ and δ are the dimensions of h^m_{BOS} and $\hat{n}^m_{\mathcal{S}}$, respectively. We concatenate and feed the result into a feedforward network to obtain the integrated feature $z \in \mathbb{R}^{\epsilon+\delta}$. Hence, z is a feature after the fusion of textual and structured information. Afterward, we split z into two parts as the updated features $\tilde{h}^m_{BOS} \in \mathbb{R}^\epsilon$

and $\tilde{n}_{\mathcal{S}}^{m} \in \mathbb{R}^{\delta}$ of the [BOS] token and *Super Node*, respectively.

$$z = \mathrm{GeLu}(W([h_{BOS}^{m}; \hat{n}_{\mathcal{S}}^{m}]) + b) \quad (5)$$

5.4 Event Type Classifier

After updating the features through M integration layers, the super node's feature $\tilde{n}_{\mathcal{S}}^{M}$ and the mean pooling result θ^{M} of graph $G^{D,M}$ are concatenated with the hidden state of [BOS] token to obtain the feature h for the event type identification. We use different classifiers to identify the event type of the event triple from different stories. For the events in $\mathcal{U}$, we use the sigmoid function following a feedforward network ϕ to identify whether it is *Forgotten* or *Unforgotten*. And the loss is denoted as $\lambda_{\mathcal{U}}$. Otherwise, we apply the softmax function following another feedforward network ψ to determine whether the event in $\mathcal{V}$ is *Consistent*, *Inconsistent*, or *Additional*. And the loss is denoted as $\lambda_{\mathcal{V}}$.

$$h = \tilde{h}_{BOS}^{M} \oplus \tilde{n}_{\mathcal{S}}^{M} \oplus \theta^{M} \quad (6)$$

$$y_i^{D'} = \begin{cases} \mathrm{Sigmoid}(W_\phi h + b_\phi) & D' = \mathcal{U} \\ \mathrm{Softmax}(W_\psi h + b_\psi) & D' = \mathcal{V} \end{cases} \quad (7)$$

5.5 Related Node Classifier

To extract the support evidence, we identify whether the node N_j in G^{D} is related to $e_i^{D'}$. Thus, each node feature is fed into a feedforward network following a sigmoid layer to perform binary classification. Note that *Forgotten* and *Additional* are the events only occurring in D'. The related nodes cannot be found in D. Thus, we exclude these two events to train the related node classifier, and the loss is denoted as $\lambda_{\mathcal{G}}$. Finally, we compute the weighted sum of three losses as shown in Equation 9 to update the model, where α and β are 0.5 after tuning by the validation set.

$$y^{N_j} = \mathrm{Sigmoid}(\hat{n}_j^{M} \cdot W) \quad (8)$$

$$\lambda = \alpha \cdot (\lambda_{\mathcal{U}} + \lambda_{\mathcal{V}}) + \beta \cdot \lambda_{\mathcal{G}} \quad (9)$$

6 Comparison with Natural Language Inference

To identify event types, we propose a pilot model to determine the relations between $e^{D'}$ and D. This is different from simply comparing the relation between two sentences in a natural language inference (NLI) task (Bowman et al., 2015; Williams et al., 2018; Camburu et al., 2018). Identifying the event types in narratives involves two main challenges. Firstly, the event type of the event in D' must be determined by identifying the event pair relations with all life events in D, since the discourse structures in D and D' are often different. Secondly, the granularity of event descriptions between the stories in a pair can differ. Hence, to determine the event type, we must infer the relevant details of the described events in both stories.

To investigate the difference between NLI and event type identification in information recall assistance, we experiment with the impact of introducing the NLI task into our model. We find that pre-training the language model on the NLI task and fine-tuning the model on our task will improve the performance. However, the label definitions in the NLI task are different from our task. The details are discussed in Appendix D.2.

7 Experiments

7.1 Baseline Models

Since the stories in our dataset are lengthy, we exploit the models that are capable of encoding the whole story as our baseline models.

XLNet (Yang et al., 2019): XLNet is a sequence-to-sequence autoregressive model that pre-trains with the permutation language modeling task instead of the masked language model task in BERT (Devlin et al., 2018). To determine the event type, we use the hidden state of the last [EOS] token as input of the event type classifiers.

GPT-2 (Radford et al., 2019): In addition to the model equipped the autoencoder, we fine-tune an autoregressive model–GPT-2 on our dataset for event type identification.

BART (Lewis et al., 2019): BART is a sequence-to-sequence model that can encode lengthy documents. Compared with our model only containing the autoencoder, BART consists of an autoencoder and autoregressive decoder.

Longformer (Beltagy et al., 2020): Since the number of the story tokens exceeds 512, we utilize Longformer which is capable of encoding long-lengthy documents. To fine-tune the model, we concatenate $e^{D'}$ and D as the input, and use the classifiers mentioned in Section 5.4.

Longformer with GATs: To encode the event graphs, we simply stack M layers of GAT into Longformer. The final hidden states of Longformer and the GAT layer are concatenated and input to the classifiers for identify the event type. The related node classifier is included.

Model	Overall	Consistent	Inconsistent	Additional	Forgotten	Unforgotten
XLNet	0.6062	0.7076	0.0238	0.7911	0.7925	0.7159
GPT2-large	0.6025	0.6999	0.0000	0.8063	0.7955	0.7107
BART-large	0.6369	0.7582	0.0247	0.8324	0.8135	0.7555
Longformer-base	0.6183	0.7340	0.0000	0.8256	0.8081	0.7237
Longformer-large	0.6334	0.7462	0.0142	0.8315	0.8221	0.7529
Longformer-large w/ GATs	0.6531	0.7472	0.1095	0.8337	0.8158	0.7591
GreaseLM-like (Longformer-large)	0.6351	0.7592	0.0000	0.8354	0.8221	0.7589
SEEN (BART-large)	0.6384	0.7623	0.0000	0.8385	**0.8268**	0.7641
SEEN (Longformer-base)	0.6341	0.7379	0.0550	0.8183	0.8120	0.7471
SEEN (Longformer-large)	**0.6654**	**0.7633**	**0.1313**	**0.8411**	0.8262	**0.7653**

Table 1: Experimental results of detecting information recall needs.

Model	F-score	Precision	Recall
Longformer-large w/ GATs	0.7289	0.7360	0.8304
SEEN (Longformer-large)	**0.7888**	**0.7775**	**0.8883**

Table 2: Results of support evidence extraction task.

7.2 Experimental Setup

The story pairs in our dataset are randomly split into training, validation, and test sets by the ratio 8:1:1. In other words, the training, validation, and test sets consist of 1,002, 48,413, and 129 pairs of stories, and 48,413, 5,913, and 6,563 events, respectively. In our experiments, we exploit several language models to evaluate the performance of encoding textual data. To load the pre-trained weight of the language model in our proposed model–SEEN, the sum of L and M is the same as the number of the transformer layers in the original language model. In SEEN, we have experimented with M ranging from 3 to 8. The detail of the comparison is described in Appendix D.3. And the contributions of different layers are reported in Appendix D.4. Finally, we report the results of setting M as 5 in the following sections. To ensure reliability, we train each model three times with different seeds and report the average performance.

7.3 Experimental Reuslts

The performance of each model on overall event types are shown in Table 1. We also report the results of each event type. F-score is adopted as the evaluation metric. We calculate McNemar's statistical significance test on the baselines and our models. To verify the effectiveness of the integration layer, we compare the performances of the following three combinations: (1) "BART-large" and "SEEN (BART-large)". (2) "Longformer-base" and "SEEN (Longformer-base)". (3) "Longformer-large" and "SEEN (Longformer-large)". The performances of SEEN in the three combinations outperform the baseline models at $p < 0.01$, $p < 0.05$, and $p < 0.01$, respectively. The results show the adaptability of the integration layer to different language models.

We find that "Longformer-large w/ GATs" significantly outperforms all the other baselines. That means incorporating the event graph is able to encode event relations to improve the performance. In addition, training the task of support evidence extraction simultaneously benefits the performance of event type identification. Moreover, "SEEN (Longofrmer-large)" outperforms "Longformer-large w/ GATs", suggesting that our proposed fusion mechanism introduces structured information effectively to enhance the language model. Comparing the last three rows, the Longformer-based encoder is better than the BART-based, and "SEEN (Longofrmer-large)" achieves the highest overall performance. The reason may be that the integration layers are built on the encoder layer. Identifying the event types by exploiting the output of the hidden states from the integration layer connected with the autoencoder is more suitable for our task. While the prediction of "SEEN (BART-large)" is based on the hidden states output from the autoregressive decoder. Note that all the models achieve relatively lower scores on *Inconsistent* type because the number of this event is sparse in NIR. Besides, we find that "SEEN (Longofrmer-large)" usually identifies *Inconsistent* as *Additional*. The further error analysis is shown in Appendix D.1.

To verify the impact of the integration layer on the support evidence extraction task, we compare our proposed model SEEN with "Longformer-large w/ GATs" which simply concatenates the hidden states of the language model and the GAT layer. The evaluation metric is macro-averaged F-score. As mentioned in Section 5.5, we only extract the related nodes of *Unforgotten*, *Consistent*, and *Inconsistent* events. In Table 2, SEEN

Model	F-score
SEEN (Longformer-large)	**0.6654**
w/o pre-training on NLI	0.6488
w/o Concat	0.6408
w/o Support Evidence Extraction	0.6349
w/o Event Graph	0.6334

Table 3: Ablation study of SEEN.

Correctness of Detection	F-score of Extraction
Correct	0.8095
Wrong	0.6671

Table 4: Relation between Detecting Information Recall Needs and Extracting Support Evidence

outperforms "Longformer-large w/ GATs". That means fusing the textual and structured information improves the related node selection. In addition, to compare SEEN with GreaseLM, we re-implement GreaseLM with the best setting, which is referred to as GreaseLM-like (Longformer-large). The slight difference is that node representations are built from language models. Since most of the nodes in our event graph are text spans, we cannot leverage existing node embeddings. We also report the comparison of SEEN and the GreaseLM-like model in Table 1. The result shows that SEEN outperforms the GreaseLM-like model, suggesting the robustness of our proposed integration layer.

8 Analysis and Discussion

8.1 Ablation Study

In this section, we perform an ablation study to analyze the impact of SEEN with different settings.
w/o pre-training on NLI: We introduce the NLI task to strengthen the ability of our language model on capturing semantic features to infer the consistency of event descriptions. Hence, we investigate the influence of pre-training the language model on the Multi-NLI (Williams et al., 2018) dataset.
w/o Concat: Instead of concatenating the final hidden state of the super node and the average of node representations, we only use the hidden state of `[BOS]` as the input of the event type classifier to evaluate the importance of structured features.
w/o Support Evidence Extraction: To analyze the impact of extracting support evidence toward the event type identification, we construct a classifier to extract related nodes in the event graph as evidence for explaining the event type predictions.
w/o Event Graph: To investigate whether the structured event information is beneficial for capturing the fine-grained relations between life events, we analyze the impact of with or without event graphs on the task of detecting information recall needs. Specifically, "SEEN w/o Event Graph" is the alias of the baseline model "Longformer", which does not encode the event graph.

The ablation study results are shown in Table 3. We find that the performance degrades the most when the event graph is excluded, suggesting that enhancing the structured event information to the language model benefits the event type identification results. In addition, introducing the subtask of extracting support evidence into SEEN can also assist the model in detecting information recall needs. Furthermore, pre-training on the Multi-NLI dataset and fine-tuning on our NIR dataset is also beneficial for identifying the semantic relatedness between $e^{D'}$ and D. We further perform an experiment to analyze the relevance between the NLI task and the task of detecting information recall needs. Experimental results, reported in Appendix D.2, show that the NLI task is different from our task.

8.2 Case Study of Support Evidence Extraction

To investigate the result of the support evidence extraction task, we perform the case study and plot the selected nodes as shown in Figure 4. Case (a) is an *Inconsistent* event since the host of the party described in the event sequence and the event graph (constructed from the reference story) are different. In this case, most of the selected nodes are correct, which are related to the described event and can explain why the event is inconsistent. In contrast to case (a), case (b) fails to select the related nodes and the prediction of the event type is also incorrect. Although the model selects all related nodes in case (c) correctly, the prediction of the event type is wrong. Here, the caller of 911 is the author, not the others, while SEEN classifies the *Inconsistent* event as *Additional*. Note that even though case (c) shows the event type identification is incorrect, SEEN is still capable of reminding the user that the event is forgotten or confused by providing the related nodes. In this way, SEEN can proactively provide information recall assistance.

Furthermore, we also verify the relatedness between the tasks of detecting information recall needs and extracting support evidence. Table 4 reports the F-score of support evidence extraction depending on whether the result of detecting infor-

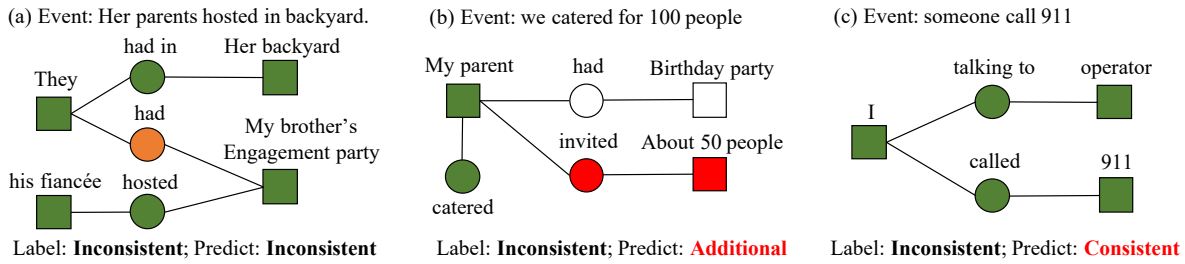

Figure 4: The examples of the support evidence extraction task. The nodes in the circle and square shapes are predicates, and entities (i.e., subjects or objects), respectively. The green nodes are the correct selections, the red nodes are ground truth but not selected, and the orange nodes are selected nodes but not ground truth.

mation recall needs is correct. In this quantitative evaluation, we find that if the need for information recall is correctly detected by SEEN, our model also achieves promising performance in support evidence extraction. On the other hand, even if the prediction of information recall need is wrong, SEEN could extract some valuable evidence to remind the user.

9 Related Work

Recently, more and more works show their interests in lifelogging. Some works have investigated lifelogging applications on lifestyle understanding (Doherty et al., 2011), diet monitoring (Maekawa, 2013), and contact tracing (Bengio et al., 2020). In addition, several studies have worked on the reactive information recall service. Gurrin et al. (2016, 2017, 2019, 2020) introduce visual lifelog retrieval tasks that aims at querying specific moments in a lifelogger's life. Chu et al. (2019) and Chu et al. (2020) construct a multimodal retrieval model that enables users to search their photos with textual queries. Yen et al. (2021b) propose a system to answer the questions about personal life experiences over personal knowledge base. In this work, we focus on detecting the need for a proactive information recall service along with the support evidences.

The structured information, such as dependency parsing results, has proved the effectiveness in capturing the contextual interactions. For instance, the model proposed by Gong et al. (2022), BERT4GCN (Xiao et al., 2021), and SGNET (Zhang et al., 2020) integrate the dependency relations to leverage syntactic information. GreaseLM (Zhang et al., 2022) and LUKE (Yamada et al., 2020) integrate the external knowledge base by fusing token representations and entity representations from the language model and the additional embeddings, respectively. Here, we introduce an event graph into our proposed model to capture the relations of the life events.

10 Conclusion

Information recall has attracted much attention in recent years. In contrast to previous studies, we present the task of proactive information recall support and construct NIR, the first human-annotated dataset, to investigate the need for information recall. In this work, we seek to detect event relations between life experiences retold at different times, and identify five event types to determine the time to trigger information recall. To identify the event types for information recall assistance, a pilot model–structured event enhancement network (SEEN) is proposed. We construct an integration layer to fuse the structured information from the event graph into textual representations. In addition, SEEN provides the support evidence to the events by selecting the related nodes in the event graph. Users can consult the explanation to recall their past experiences. However, identifying inconsistent events is still challenging; this is left as future work. We also plan to construct an end-to-end system to extract life events in narratives and provide proactive information recall support. Besides, at the current stage, we utilize the gold event graph of each story in our experiments. In the future, we will explore the method to extract personal life events from document to construct a personal knowledge graph.

Acknowledgements

This research was partially supported by National Science and Technology Council, Taiwan, under grants MOST 110-2221-E-002-128-MY3 and MOST 110-2634-F-002-050-.

11 Limitations

As time passes, many events continuously happen in our daily life. Consulting only one document that describes personal life experiences is not enough to identify the need for information recall assistance in the real-world application. However, the dataset that can be applied to investigate the issue of detecting information recall needs is hard to collect. We extend the Hippocorpus dataset, whose nature is in line with our work, to construct the NIR dataset. On the other hand, although our NIR dataset provides two versions of stories of the same events written at different times, we still cannot confirm which story, the previous one or the latter, is correct when contradictory. In this work, we propose a pilot exploration of proactively information recall assistance. To this end, we simply postulate that the story written at the previous time was correct when the user was still deeply impressed by the life events, so the story written at that time was used as a reference story. In addition, the number of inconsistent events is relatively lower in our dataset due to the human writing habit of avoiding uncertain events. In other words, when writing a diary, we always write the ones we exactly remember, which leads to difficulty collecting inconsistent events.

12 Ethics Statement

Considering the potential infringement of privacy in the lifelog research, this section is an ethics-related elaboration for our dataset collection and a statement to address the risk of ethics for the methods. Our dataset "NIR" is an extension of an existing public dataset "Hippocorpus". The Hippocorpus dataset is collected from the crowdsourcing that the workers were to write the stories and the summaries twice at different times, and the other workers were to write the imagined version of the stories based on the summaries. The demographic information (age, gender) is optionally reported by the workers. However, the workers' IDs and names are not included in the Hippocorpus dataset. In other words, the dataset does not contain any personally identifiable information that would infringe on someone's privacy. In this work, we will only release the life event annotation and the support evidence of the event types in stories for research purposes. The stories in the Hippocorpus dataset will not be included in NIR. Hippocorpus can be accessed from the website. [2] However, in the real world, lifelog applications could suffer from the risk of personal information leakage. The misuse of data and BAD (Broken As Designed) systems may violate the regulation or laws on data protection and privacy (GDPR, etc.). Hence, we leave the investigation of a privacy-aware lifelogging framework as future work.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Yoshua Bengio, R. Janda, Y. W. Yu, Daphne Ippolito, Max Jarvie, D. Pilat, Brooke Struck, Sekoul Krastev, and A. Sharma. 2020. The need for privacy with public digital contact tracing during the covid-19 pandemic. *The Lancet. Digital Health*, 2:e342 – e344.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9539–9549. Curran Associates, Inc.

Tai-Te Chu, Yi-Ting Liu, Chia-Chung Chang, An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Nlp301 at the ntcir-15 micro-activity retrieval task: incorporating region of interest features into supervised encoder. In *Proceedings of the NTCIR-15 Conference*.

Tzu-Hsuan Chu, Hen-Hsen Huang, and Hsin-Hsi Chen. 2019. Image recall on image-text intertwined lifelogs. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 398–402. IEEE.

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*.

Aiden R Doherty, Niamh Caprani, Vaiva Kalnikaite, Cathal Gurrin, Alan F Smeaton, Noel E O'Connor, et al. 2011. Passively recognising human activities

[2] http://aka.ms/hippocorpus

through lifelogging. *Computers in Human Behavior*, 27(5):1948–1958.

Charles J Fillmore, Miriam RL Petruck, Josef Ruppenhofer, and Abby Wright. 2003. Framenet in action: The case of attaching. *International Journal of Lexicography*, 16(3):297–332.

Zheng Gong, Kun Zhou, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2022. Continual pre-training of language models for math problem understanding with syntax-aware memory network. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5923–5933.

Cathal Gurrin, Hideo Joho, Frank Hopfgartner, Liting Zhou, and Rami Albatal. 2016. NTCIR Lifelog: The first test collection for lifelog research. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–708.

Cathal Gurrin, Hideo Joho, Frank Hopfgartner, Liting Zhou, Rashmi Gupta, Rami Albatal, and Duc Tien Dang Nguyen. 2017. Overview of ntcir-13 Lifelog-2 task.

Cathal Gurrin, Hideo Joho, Frank Hopfgartner, Liting Zhou, V-T Ninh, T-K Le, Rami Albatal, D-T Dang-Nguyen, and Graham Healy. 2019. Overview of the ntcir-14 Lifelog-3 task. In *Proceedings of the 14th NTCIR Conference*, pages 14–26. NII.

Cathal Gurrin, Tu-Khiem Le, Van-Tu Ninh, Duc-Tien Dang-Nguyen, Björn Þór Jónsson, Jakub Lokoš, Wolfgang Hürst, Minh-Triet Tran, and Klaus Schoeffmann. 2020. Introduction to the Third Annual Lifelog Search Challenge (lsc'20). In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 584–585.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

Pei-Wei Kao, An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2021. Convlogminer: A real-time conversational lifelog miner. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4992–4995. International Joint Conferences on Artificial Intelligence Organization. Demo Track.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Takuya Maekawa. 2013. A sensor device for automatic food lifelogging that is embedded in home ceiling light: A preliminary investigation. In *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, pages 405–407. IEEE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Maarten Sap, Eric Horvitz, Yejin Choi, Noah A. Smith, and James Pennebaker. 2020. Recollection versus imagination: Exploring human memory and cognition via neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1970–1978, Online. Association for Computational Linguistics.

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6578–6588, Online. Association for Computational Linguistics.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238, Online. Association for Computational Linguistics.

Yu-Wun Wang, Hen-Hsen Huang, Kuan-Yu Chen, and Hsin-Hsi Chen. 2018. Discourse marker detection for hesitation events on mandarin conversation. In *Interspeech*, pages 1721–1725.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Zeguan Xiao, Jiarun Wu, Qingliang Chen, and Congjian Deng. 2021. BERT4GCN: Using BERT intermediate layers to augment GCN for aspect-based sentiment classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9193–9200, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2019. Personal knowledge base construction from text-based lifelogs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 185–194.

An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Multimodal joint learning for personal knowledge base construction from twitter-based lifelogs. *Information Processing & Management*, 57(6):102148.

An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2021a. Ten questions in lifelog mining and information recall. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, pages 511–518.

An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2021b. Unanswerable question correction in question answering over personal knowledge base. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations*.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. Sg-net: Syntax-guided machine reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9636–9643.

A Life Event Annotation

As Section 3.1 mentioned, we follow the definition in LiveKB (Yen et al., 2019, 2020) and annotate each event with polarity, explicit and implicit. In an explicit event, the predicate can be annotated by directly using the words in the story. In an implicit event, the predicate must be inferred from the context since the action of the event is not mentioned in the story. For implicit predicates, annotators were to choose the proper predicate by consulting FrameNet (Fillmore et al., 2003). For instance, two explicit events *(She, hosted, party)* and *(She, hosted in, backyard)* are included in Sentence (2). A single implicit life event *(I, drink, the shots and cocktails)* is described in Sentence (3).

For the life event annotation, we invited five annotators who majored in linguistics or were English native speakers. Given a story, the annotators were to annotate life events in the story in triple form. To verify the quality of the annotation results, we sampled five stories (i.e., a total of 100 sentences and 129 life events) as reference story and asked a supervisor to label the life events. These stories were also assigned to the other four annotators. Since the three components in the triple were annotated as free text, we joined each component into a sequence.

We measured the agreement of each annotator with the supervisor via the Rouge-L (Lin, 2004) and F-scores for the life event triple and the explicitness of the life event, respectively. Here, the reason for utilizing the Rouge-L score to evaluate the agreement of life event triple annotation is that the components in a triple are text spans. We regard the annotation results of the supervisor as the reference to measure the annotation quality of the other annotators. The resulting average agreement of the life event triple and the explicitness of the life event were 0.87 and 0.80, respectively.

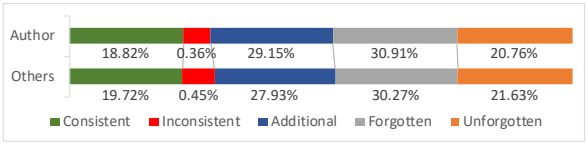

Figure 5: The ratio of event type on people recalling their own and others' events.

B Event Type Analysis on Ownership

Note that people recall not only their life events but also events involving family, friends, and acquaintances. We further investigated whether people tended to remember their own experiences better than those of others. At the current stage, as events are not labeled to indicate to whom the event belongs, we classified events that do not contain the

Hyperparameter	SEEN (BART-large)	SEEN (Longformer-base)	SEEN (Longformer-large)
Parameter Size	529M	221M	556M
Number of Integration Layers	5	5	5
Number of attention heads in GNN	16	16	16
Dimension of node feature in GNN	1024	768	1024
Dropout rate in GNN	0.2	0.2	0.2
Learning rate	1e-5	1e-5	1e-5
Number of epoch	5	5	5
Optimizer	AdamW	AdamW	AdamW

Table 5: Hyperparameter of each model.

Model	Average training time (hr/epoch)
SEEN (BART-large)	0.47
SEEN (Longformer-base)	0.36
SEEN (Longformer-large)	0.87

Table 6: Time consumption to train the models.

words "I", "me", "we", or "us" in the subject or object as life events belonging to others. Otherwise the event was taken to be a life event of the author.

The result is shown in Figure 5, where the bars represent the smoothed and normalized ratios. We find that the ratio of *Inconsistent* events in recalling other people's life events is higher than that when recalling their own life events. The ratio of *Additional* events is also lower when recalling other people's life events: when people write a retold story, they describe only those life events of others that they remember. Hence, when people describe life events again in a post-retold story, they rarely mention new life events about others. However, as people do not remember the life events of others as clearly as their own, they are more prone to confusing such life events.

C Details of Experimental Setup

For each hyperparameter trial, we evaluate it on the validation set, and the one with the highest score on the event type identification task will be chosen. Apart from the hyperparameters, we evaluate our methods on the validation set 10 times in each epoch. The one with the highest score will be treated as the final checkpoint and reported its test set performance. The hyperparameters of each model are reported in Table 5. In addition, we use eight V100 GPUs to train our models and report the average training time in Table 6.

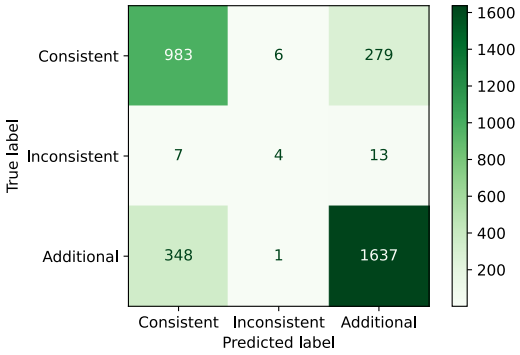

Figure 6: Confusion matrix of "SEEN (Longformer-large)" for event type identification.

D Additional Experiments and Discussion

D.1 Error Analysis

To investigate the performance of SEEN on each event type, Figure 6 shows the confusion matrix of our model in predicting *Consistent*, *Inconsistent*, and *Additional*. We find that SEEN predicts most *Inconsistent* events as *Additional* events. Firstly, although people often mix their experiences, we tend to avoid unclear events while writing, which results in the rareness of the *Inconsistent* event in our datasets. Apart from the problem of limited training data, this may be because determining that the described event conflicts with established facts require further reasoning on details such as the number of events that occurred, the order of activities, the friend's name, or the object description. Furthermore, since both *Inconsistent* and *Additional* cannot be found in the story context, it is more difficult to classify the event between these two types, which may cause misclassifying *Inconsistent* event as *Additional* event.

D.2 Impact of Pre-training Task

To further compare the event type identification task with the NLI task, we experiment the differ-

Pre-training on Multi-NLI	Fine-tuning on Post-Retold Events	Overall	Consistent	Inconsistent	Additional
v		0.4075	0.4715	0.0397	0.7113
	v	0.5480	**0.7556**	0.0625	0.8260
v	v	**0.5572**	0.7512	**0.0837**	**0.8367**

Table 7: Results of different pre-training task settings.

ent pre-training task settings. In other words, we note that the labels are different between the Multi-NLI dataset and our NIR dataset. Therefore, we align the *entailment*, *contradiction*, and *neutral* in Multi-NLI with *Consistent*, *Inconsistent*, and *Additional* in post-retold of NIR, respectively. We report the overall macro-averaged F-score and F-score of each individual label. The first two columns denote whether the methods are trained on the Multi-NLI dataset and fine-tuning on the NIR dataset to identify the event types in the post-retold stories.

As shown in Table 7, the method only trained on the Multi-NLI dataset does not work well in detecting information recall needs. That means the label definitions between NLI and NIR are marginal different, especially the *Consistent* events. We speculate the reason is that, in the "Entailment" class, most hypotheses are another way of saying the premises. However, the hypotheses are irrelevant to the premises if the relations are "Contradiction". By contrast, determining the event types of "Conststent" and "Inconsistent" requires the ability to recognize subtle differences between the descriptions, such as the sequence of several life events. SEEN trained on both datasets achieves the highest performance. It means pre-training on the NLI task helps the model better capture semantic relatedness between two descriptions.

D.3 Number of Integration Layers

We further compare the performance of SEEN with the different numbers of the integration layers. Experimental results shown in Table 8. We find that SEEN with five integration layers ($M = 5$) achieves the highest performance, which is the same as the result of GreaseLM. However, different from GreaseLM, there is no consistency in performance changes while M decreases or increases. We think the reason is that the way SEEN fuses textual and structured features are by iteratively initializing the node representations with the updated token representations in each integration layer (The process is described in Section 5.3). While GreaseLM utilizes additional node embeddings as node representations, and concatenates the parts of hidden states from the language model and the node embeddings without re-initializing the node representations.

# of Integration layer(M)	F-score
M = 3	0.6559
M = 4	0.6470
M = 5	**0.6654**
M = 6	0.6568
M = 7	0.6414
M = 8	0.6597

Table 8: Performance of different number of the integration layer.

D.4 Contribution of Different Fusion Layers

To investigate the contribution of each fusion layer in SEEN, we compute the distribution of the edge weights between nodes in the GAT layer. We denote the edges connecting to the related node and the unrelated node as E_{RN_+} and E_{RN_-}, respectively. To show the difference between the edge weights, we tell whether the edge weights are higher than the threshold (0.5). If the edge weight is higher than the threshold, the edge is denoted as the positive case as "triggered edges". In the first GAT layer, the distributions of edge weights are relatively average. That leads to none of the edges is triggered edge. This might be that the first GAT layer attempts to capture structured information of the whole event graph by gathering the messages from the neighbor nodes. In contrast, 6.74% edges are triggered edges in the last GAT layer, which is much more than those in the first layer. We further compare the triggered edge distribution of E_{RN_+} and E_{RN_-}, which are 14.93% and 4.92% in the last GAT layer, respectively. That is, compared with the first GAT layer, the last GAT layer in the integration layer aims to focus on the information related to the $e_i^{D'}$.