Focus on the Action: Learning to Highlight and Summarize Jointly for Email To-Do Items Summarization

Kexun Zhang[1], **Jiaao Chen**[2], **Diyi Yang**[2]
[1]Zhejiang University
[2]Georgia Institute of Technology
kexunz@zju.edu.cn
jiaaochen@gatech.edu
diyi.yang@cc.gatech.edu

Abstract

Automatic email to-do item generation is the task of generating to-do items from a given email to help people overview emails and schedule daily work. Different from prior research on email summarization, to-do item generation focuses on generating action mentions to provide more structured summaries of email text. Prior work either requires large amount of annotation for key sentences with potential actions or fails to pay attention to nuanced actions from these unstructured emails, and thus often lead to unfaithful summaries. To fill these gaps, we propose a simple and effective learning to highlight and summarize framework (LHS) to learn to identify the most salient text and actions, and incorporate these structured representations to generate more faithful to-do items. Experiments show that our LHS model outperforms the baselines and achieves the state-of-the-art performance in terms of both quantitative evaluation and human judgement. We also discussed specific challenges that current models faced with email to-do summarization.

1 Introduction

Automatic email to-do generation is the task of summarizing to-do items from given emails (Mukherjee et al., 2020) to help people overview overwhelming numbers of emails they receive every day (Radicati and Hoang, 2011) and schedule their daily work. Unlike prior research on emails such as generating email conversation summarization (Muresan et al., 2001; Nenkova and Bagga, 2003; Rambow et al., 2004), keyword extraction (Turney, 2000; Lahiri et al., 2016; Lin et al., 2017) and subject line generation (Zhang and Tetreault, 2019; Xue et al., 2020), to-do items generation focuses more on *action mentions* to provide more *structured* summaries from email communications, which requires identifying important tasks to be performed among all the action items and aligning them with the right users (Mukherjee et al., 2020).

To tackle these challenges in to-do item generation, a two-stage framework (Zhang and Tetreault, 2019; Mukherjee et al., 2020) is generally utilized to first extract the commitment sentence which is the most related to the to-do item and then generate to-do items based on those selected sentences. Despite the effectiveness, several limitations exist: (1) extra annotations are usually needed for the first stage (Mukherjee et al., 2020) to learn the classifiers which require extra cost/expertise and are often hard to obtain in low-resourced settings; (2) any information loss in the identification stage might lead to bigger noises in the to-do generation stage; (3) directly extracting actions from less-structured text usually leads to unfaithful summaries (Chen and Yang, 2021) that mismatch the relations between users and actions.

To fill in these gaps, in this work, we propose a learning to highlight and summarize model, where we learn the important sentence identification module and to-do summarization module concurrently in an end-to-end manner to focus on the most salient actions, as well as to incorporate structured action representations to generate more faithful to-dos. One example is shown in Figure 1. Specifically, we use an unsupervised approach to extract the salient sentences and actions by comparing them to the ground truth to-do. We extract action triplets from the text and construct an action graph to encode the structural information in the unstructured text. During training, the model learns to generate to-do items and identify highlights jointly. During prediction, the model utilizes the predicted highlights by modifying the attention distribution accordingly. Furthermore, as extrinsic hallucinations involving named entities are common in automatic summarization tasks (Maynez et al., 2020; Chen et al., 2021), we propose a perturbation technique based on person names to reduce the extrinsic hallucinations.

We conduct experiments on the SmartToDo cor-

Findings of the Association for Computational Linguistics: ACL 2022, pages 4095 - 4106

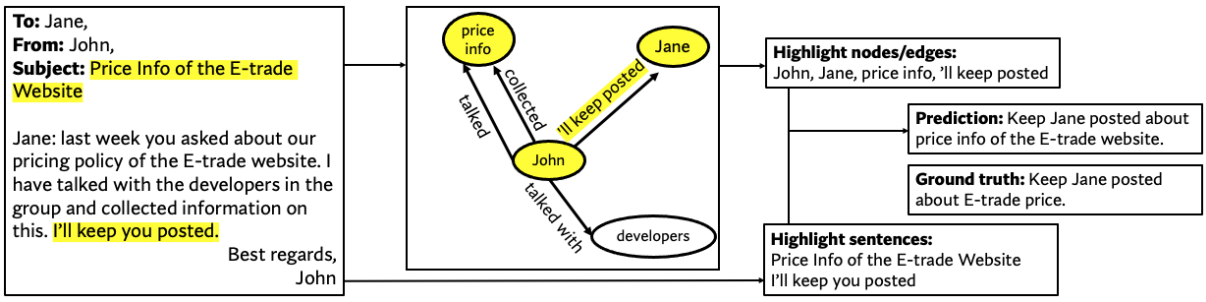


Figure 1: Example pipeline of our method to generate to-do items from email. Highlight sentences are extracted from the email text. Highlight action nodes are extracted from the constructed action graph. The highlight actions and sentences are then utilized to generate the to-do item.

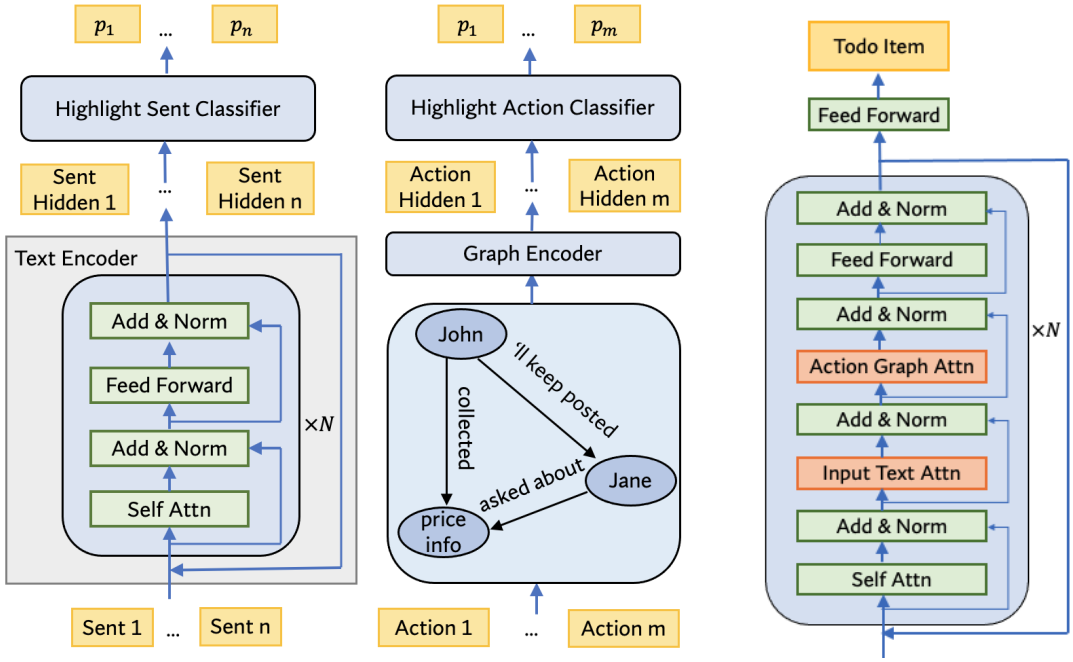
pus(Mukherjee et al., 2020) and show that our LHS model outperforms strong baselines on to-do item generation, demonstrating the effectiveness of this joint model on learning to highlight and summarize. By evaluating our LHS model on a different email corpus, we demonstrate that it generalizes well on a zero-shot condition.

2 Related Work

Document summarization Recent works on document summarization are usually extractive (Gupta and Lehal, 2010; Narayan et al., 2018; Liu and Lapata, 2019) and abstractive. Methods for abstractive document summarization include sequence-to-sequence models (Rush et al., 2015), pointer generators (See et al., 2017), reinforcement learning (Paulus et al., 2018; Huang et al., 2020) and methods based on pre-trained language models (Lewis et al., 2020; Zhang et al., 2019). To generate faithful abstractive document summaries (Maynez et al., 2020), recent works on abstractive summarization have incorporated different types of guidance signals extracted including key tokens (Gehrmann et al., 2018; Chen and Bansal, 2018; Li et al., 2018; Saito et al., 2020; Dou et al., 2021), highlight sentences (Liu et al., 2018; Dou et al., 2021; Saito et al., 2020) and relational triplets (Jin et al., 2020; Zhu et al., 2020a; Dou et al., 2021). However, these models are mainly designed to find important information from all parts of the document, while todo items are often associated with only several sentences in emails.

Conversation summarization Automatic to-do generation is also similar to the task of conversation summarization in the way that the input data contains both the current email in which the user promises to take an action and the previous email to which the user is replying. For extractive conversation summarization(Murray et al., 2005), statistical machine learning methods such as skip-chain CRFs (Galley, 2006), SVM with LDA models (Wang and Cardie, 2013), and multi-sentence compression algorithms (Shang et al., 2018) have been used. When key information is scattered in multiple sentences, such methods had trouble in succinctness, fluency and naturalness (Song et al., 2020). Abstractive conversation summarization methods are more effective in these circumstances. These methods design hierarchical models (Zhao et al., 2019; Zhu et al., 2020b), incorporate commonsense knowledge (Feng et al., 2020), or leverage conversational structures like dialogue acts (Goo and Chen, 2018), key point sequences (Liu et al., 2019a), topic segments (Liu et al., 2019b; Li et al., 2019) and stage developments (Chen and Yang, 2020). Some recent research has also utilized structural information to detect important content in conversations (Murray et al., 2006; Bui et al., 2009; Qin et al., 2017). However, recent methods that explicitly identify salient content has not been used jointly with conversation summarization models.

Summarization tasks on email Several summarization-like tasks have been proposed such as email text summarization (Muresan et al., 2001; Nenkova and Bagga, 2003; Rambow et al., 2004), keyword and action extraction (Turney, 2000; Lahiri et al., 2016; Lin et al., 2017; Corston-Oliver et al., 2004), subject line generation (Zhang and Tetreault, 2019; Xue et al., 2020) and to-do generation task (Mukherjee et al., 2020). These previous works often deploy a two-stage framework that first extracts salient sentences from the email and then generates summary based on extracted sentences. However, any information loss in the first stage might lead to bigger noises in the to-do



(a) Encoder (b) Decoder

Figure 2: Model Architecture. The sentences and actions are first encoded and then fed to the highlight classifiers. The hidden representations of sentences and actions, along with their probability of being highlights are then used in the cross-attention layer in the decoder. The email encoder has the same structure as BART encoder. The graph encoder utilizes graph attention networks to encode the action graph.

generation stage, and directly extracting actions from less-structured text might lead to unfaithful summaries (Chen and Yang, 2021). To fill these gaps, we propose to explicitly model actions within sentences in the email in a structured way, by using action graphs to model the actions, introducing an auxiliary classifier to identify the highlights, and further combining these with a decoder that attends to both the encoding of input tokens and extracted actions for to-do generation.

3 Methods

To generate faithful to-do items that correlate actions with correct users from emails[1], we propose to encode both text-level and action-level information as guidance signals. The architecture of LHS model is shown in Figure 2. We first extract the action mention ("WHO-DOING-WHAT" triples (Chen and Yang, 2021)) from the emails and construct an action graph (Section 3.1), then we encode text and actions separately (Section 3.2) and use a multi-scale decoder that attends to both email and action graphs to generate the to-do item (Section 3.4). Besides directly incorporating different levels of information, we further design an end-to-end framework to utilize guidance signals such as important sentences and actions to help summarization models focusing more on the key to-dos by jointly learning a highlight classifier (Section 3.3) with the summarization model and view the predicted results as priors to attention distributions in the decoder (Section 3.4).

3.1 Action Graph Construction

Unlike general summarization over emails, to-do item summarization is more action-focused, requiring identification of specific tasks to be performed (Mukherjee et al., 2020). As a result, we propose to explicitly inject structured action information into the to-do summarization model. To do so, we first extract actions triples from the emails in the form of subject-predicate-object triplets and then construct the action graphs to aid the to-do summarization.

Action extraction Following Chen and Yang (2021), we first pre-process the emails to a third-person point-of-view via (1) replacing first-person pronouns with the name of the sender of the email,

[1]Before generating to-do items for a given email, we must first tell whether it contains a to-do item. This problem is addressed by (Mukherjee et al., 2020) using an RNN-based classifier during the construction of the SmartToDo dataset, hence we focus on the generation problem in our paper.

(2) replacing second-person pronouns with the each of the names of the recipients of the email, (3) replacing the third-person pronouns with the coreference resolution algorithm provided by Stanford CoreNLP (Manning et al., 2014). Then we utilize OpenIE (Angeli et al., 2015) to extract subject-predicate-object triplets from the processed email.

Action graph construction With the extracted triples, we construct the action graph $G = (\mathbf{V}, \mathbf{E})$, where the nodes in $\mathbf{V}$ represent the subjects, predicates and objects in the action triplets, and $\mathbf{E}_{ij} = 1$ if nodes i and j appear in one same triplet. Note that we construct one action graph per email. One example is shown in Figure 1, where the subjects and objects of the actions triplets are 'John', 'Jane', 'developers' and 'price info', and the predicates of the actions include 'talked', 'talked with', 'will keep posted' and 'collected'.

3.2 Encoder

Our LHS model includes two encoders that encode the email text and the action graphs separately.

3.2.1 Email Encoder

We initialize the email text encoder E_T with a pre-trained transformer-based encoder (BART (Lewis et al., 2020)). For an input email with n sentences, $\{x_1, x_2, \ldots, x_n\}$, we concatenate and encode them together into hidden representations h_i: $\{h_1, h_2, \ldots, h_n\} = E_T(\{x_1, x_2, \ldots, x_n\})$

We truncate the email text if it exceeds the length limit of BART.

3.2.2 Action Graph Encoder

Inspired by Huang et al. (2020); Chen and Yang (2021), we utilize Graph Attention Network (Veličković et al., 2018) to encode the constructed action graph $G = (\mathbf{V}, \mathbf{E})$. Each node v with text $\{x_1^v, x_2^v, \ldots, x_k^v\}$ is initialized with the average of the hidden states from the text encoder E_T using the text of the node: $h_v = \frac{1}{k}\sum_{i=1}^{k} E_T(x_k)$.

In each layer of the graph attention network, the hidden state of every node v_i is represented by a weighted average of its neighbors' previous hidden states ($\mathbf{W}$ is a trainable parameter, $\mathcal{N}_i$ is the set of neighbors of i, and σ is the activation function):

$$\alpha_{ij} = \text{softmax}((\mathbf{W}v_i)^T(\mathbf{W}v_j))$$
$$h_i^a = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{W}v_j)$$

3.3 Highlight Identification

Guiding an abstractive summarization model with extracted salient sentences has been used in previous works on document summarization (Gehrmann et al., 2018; Chen and Bansal, 2018). Compared to typical document summarization task which needs to consider information from all parts of the document, to-do generation usually needs to focus only on a few important sentences of the email. Thus, we jointly learn an auxiliary highlight classifier to indicate whether a sentence/action is important or not. The predicted probabilities are incorporated during the decoding stage to guide the decoder paying higher attentions to the important sentences/actions in the email.

Highlight label extraction For an email $X = \{x_1, \ldots, x_n\}$ with n sentence, and its ground truth to-do item y, we automatically extract the k most important sentences for classifier learning through oracle extraction using a greedy search algorithm: (1) Rank the sentences $\{x_1, \ldots, x_n\}$ based on ROUGE scores with the ground truth y. (2) For the sentences with the k largest ROUGE scores, we label the tokens in them as highlight tokens.

Similarly, we extract the highlight actions by concatenating the triplets and calculating ROUGE scores with regard to the ground truth action, which is extracted from the concatenation of the name of the sender and the to-do item (e.g. if the sender is 'John' and the to-do item is 'Keep Jane posted about price info', the ground truth actions are extracted from the sentence 'John keep Jane posted about price info'.) The extracted actions and sentences are used only as labels to train the highlight classifiers. During inference, the LHS model directly utilizes the predictions from highlight classifiers as guidance signal.

Highlight classifier There are two highlight classifiers in the model, one for highlighting sentences and the other for highlighting actions. We utilize the hidden representation of sentences and actions given by the encoder as the input for highlight classifiers. The hidden representations are fed to a multi-layer perceptron to be classified. The classification loss is calculated with respect to the labeled highlights using cross entropy loss, where h are the real highlights and $\hat{h}$ are the predictions:

$$L_{classification}(h, \hat{h}) = -\sum_i h_i \log(\hat{h}_i)$$

We train LHS model in a joint fashion. Here $L_{classification}$ is the sum of losses for both highlight classifiers, and $L_{generation}$ is the cross entropy loss for finetuning BART for generation tasks. We use α and β as the weights to control the learning speed for two modules:

$$L_{total} = \alpha \cdot L_{generation} + \beta \cdot L_{classification}$$

3.4 Decoder

We improve the decoder of BART (Lewis et al., 2020) with additional layers of cross attention on graphs and incorporating the guidance of highlights. Each decoder layer is composed of 3 attention layers: the self-attention layer that attends to the previously generated tokens, the cross-attention layer that attends to the hidden representation of each input token, and the cross-attention layer that attends to the hidden representation of each node in the action graph. When calculating encoder-decoder attention, we modify every attention distribution with the predictions from the highlight classifiers.

Specifically, for a sequence of length l, given the original attention weights of an attention head $\alpha_{1,\dots,l}$ and the probability sequence $p_{1,\dots,l}$ where p_i denotes the predicted probability for the i-th token being highlighted (we use the probability of one sentence being highlighted as the probability for all the tokens in that sentence being highlighted). We use the L1-normalization of the predictions as the weights of modification:

$$w_i = \frac{p_i}{\sum_{j=1}^{l} p_j}$$

We then modify the attention weights from cross attentions to obtain the new distribution: $\alpha'_i = w_i \alpha_i$. The re-weighting of attention probabilities puts more weight on the highlighted sentences and actions, which makes it easier for the model to focus on the text snippets related to the to-do item.

3.5 Perturbation

According to previous studies on hallucinations (Maynez et al., 2020), about 70% of hallucinations that happen in summarization tasks, regardless of the model used, are extrinsic (model generations that ignore the input document). Among these extrinsic hallucinations, a large fraction (35%) happen on named entities (Chen et al., 2021). Such finding is consistent with our observations on this email to-do item generation task.

To address the hallucinations on named entities and prevent the model from sampling a random name from the dataset, we use a simple yet effective perturbation approach by replacing the persons' names. For each data point, we first cluster the names so that the different forms of a same name are put in the same cluster. Since the email text is preprocessed with coreference resolution as mentioned in (Section 3.1), we do the clustering by assuming two names with at least one word in common are the same.

For each cluster of names, we replace the occurrences in the cluster with a new name taken from common American names for both email text and the labeled to-do item. During training, we use original data for the first 4 epochs of fine-tuning and we use perturbed data for the last 3 epochs.

4 Experiments

4.1 Datasets

We train our LHS model on the dataset SmartToDo (Mukherjee et al., 2020). The dataset consists of around 10,000 emails and their corresponding annotated todo items. The SmartToDo dataset is constructed based on the Avocado Email corpus [2]. We also randomly sample 50 emails from the Enron Email corpus [3] and annotate them manually, in order to examine the model's performances on out-of-domain generalization.

4.2 Baselines

We compare our methods with several baselines:

- **SmartToDo** (Mukherjee et al., 2020) is the first model to address the todo generation task. We take the evaluation metrics directly from SmartToDo (Mukherjee et al., 2020) since the setting of our experiments is similar to it.

- **Transformer** (Vaswani et al., 2017): We trained transformer using OpenNMT (Klein et al., 2017) with its default setting on summarization tasks.

- **BART** (Lewis et al., 2020) is a pre-trained model that shows great performance when being fine-tuned on summarization tasks.

Dataset	Model	ROUGE-1	ROUGE-2	ROUGE-L
SmartToDo	Transformer	0.373	0.156	0.387
	SmartToDo‡	0.600	0.410	0.630
	BART	0.635	0.431	0.669
SmartToDo	LHS without highlight actions and perturbation†	0.673	0.469	0.685
	LHS without perturbation†	0.681	**0.472**	**0.692**
	LHS†	**0.686**	0.470	**0.692**
EnronEmail	BART	0.517	0.357	0.536
	LHS without perturbation†	0.573	0.412	0.589
	LHS without highlight actions and perturbation†	0.535	0.402	0.571
	LHS†	**0.592**	**0.418**	**0.614**

Table 1: ROUGE scores for different models on the SmartToDo test set and a manually annotated Enron Email dataset. ‡ indicates the current state-of-the-art method. Models with † are our proposed models.

Type	Fac.	Succ.	Inf.
Ground Truth	4.13	4.25	4.14
BART	3.87	3.95	4.02
LHS †	**3.93**	**4.08**	**4.10**

Table 2: LHS † is our learning to highlight and summarize model that incorporates highlight sentences, highlight actions and perturbation. Average human evaluation of ground truth, baselines and LHS in terms of **Fac**tualness, **Succ**intness and **Inf**ormativeness.

4.3 In-Domain Results

Quantitative evaluation We evaluate all the models with the commonly used metric for text generation, ROUGE scores (Lin and Och, 2004). We report the F1 scores of all the models in Table 1. We found that: given the large corpus used for pretraining, finetuned BART (Lewis et al., 2020) is a strong baseline that already outperforms the original SmartToDo (Mukherjee et al., 2020) model. Generally, our method that incorporates highlight sentences, action graph and perturbation outperforms the baselines greatly. Compared to the baseline model, BART, the quantitative evaluation of our method is increased by 5.1% in ROUGE-1, 3.9% on ROUGE-2 and 6.2% on ROUGE-L.The model that utilizes both highlight sentences and structured information in the action graph outperforms the model with the highlight sentences only by 1.2% in ROUGE-1, 0.3% in ROUGE-2 and 0.7% in ROUGE-L, indicating that the effectiveness of action graph and highlight action classifier. We describe some randomly picked generation examples in the Appendix.

Human evaluation We also conducted human evaluation to further assess the generation results. We randomly sampled 50 emails from the test set of the models and asked human evaluators (undergraduate research assistants who are familiar with the task) to rate the ground truth, the predictions of BART and the predictions of the proposed LHS model using a scale from 1 to 5 in three aspects: **factualness** (e.g. selecting the correct objects of the to-do action), **succinctness** (e.g. does not contain redundant information) and **informativeness** (e.g. contains the necessary information in a to-do item for the sender of the email). The Fleiss' Kappa for the raters was 0.42, showing moderate agreement (Landis and Koch, 1977). We report the results of human evaluation in Table 2. The results show that our method outperforms the baseline BART in all aspects of qualitative measures. However, our method is still weaker than human, especially in terms of factualness and succinctness, suggesting that current models still struggle with to-do item generation.

4.4 Out-of-Domain Results

To evaluate the generalizability of the proposed LHS model, we train the models on the SmartToDo dataset and evaluate them on a small batch of the Enron Email corpus which we annotated manually. The results are shown in Table 1. Given that the Enron Corpus is constructed using the emails of an energy company while the Avocado Corpus is

[2]https://catalog.ldc.upenn.edu/LDC2015T03
[3]https://www.cs.cmu.edu/ enron/

Extraction	ROUGE-1	ROUGE-2	ROUGE-L
Random	0.629	0.424	0.653
Highlight	**0.673**	**0.469**	**0.685**

Table 3: ROUGE scores of our method with randomly sampled sentences instead of highlight sentences.

Graph	ROUGE-1	ROUGE-2	ROUGE-L
Random	0.634	0.429	0.651
Action	**0.686**	**0.470**	**0.692**

Table 4: ROUGE scores of our method with action graphs and randomly constructed graphs.

from a technology company, the frequently mentioned entities and terms are different. Still, the LHS model achieves 0.592 in ROUGE-1, 0.418 in ROUGE-2 and 0.614 in ROUGE-L, indicating that LHS model generalizes well on a zero-shot condition. Furthermore, we observed that incorporating highlight sentences and actions increase ROUGE scores significantly by 7.5% in ROUGE-1, 6.1% in ROUGE-2 and 7.8% in ROUGE-L, indicating that learning to highlight and summarize jointly is also effective when generalized.

4.5 Ablation Studies

To verify the effectiveness of our proposed approaches comprehensively, we conducted a set of ablation studies as follows.

Effectiveness of highlight sentences We first examine the quality of the extracted highlight sentences and their influences on the LHS model for the to-do generation task in Table 3. We compared the ROUGE scores of the LHS model using the highlight sentences extracted by the algorithm in Section 3.3 and randomly sampled sentences. For each data point, we sample a score for each sentence from a normal distribution and select the sentences with the largest k scores as the substitute for highlight sentences, where k is the number of actual highlight sentences in that data point. The results showed that randomly sampled sentences did not improve the model's performance, and our extracted highlights were effective in guiding the model to focus on the to-do item related sentences.

Number of highlighted Sentences We also conducted experiments to show how the number of extracted highlight sentences effected the performance of the model. The results shown in Figure 3 indicates that increasing the number of highlighted sentences does not necessarily improve the model's performance. Specifically, when k increases from 1 to 3, ROUGE scores also increase; after k=3, adding more highlighted sentences seems to be associated with decreased performances. This trend makes a lot of sense as the to-do items are usually related to only 2 to 3 important sentences in the email. This pattern can also be observed in the samples shown in Table 6.

Graph	R-2 DOINGs	R-2 WHATs
LHS w/o perturbation	**0.378**	0.562
LHS	0.370	**0.593**

Table 5: ROUGE-2 scores of LHS with and without perturbation on actions extracted from the subset of the test set that contains person names. DOINGs and WHATs are the second and third part of an action triplet. WHATs in this case contain the person names.

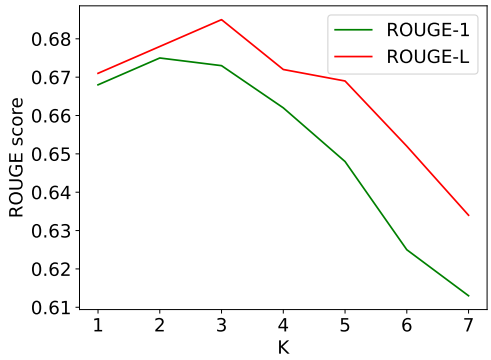


Figure 3: ROUGE scores when varying the value of k in the top k sentences extracted as highlights.

Effectiveness of action graph To show that the constructed action graph was giving guidance to the model, we conducted experiments with random graphs whose nodes are the same as action graphs but whose edges are randomly sampled. Table 4 indicates that random graphs without structural information did not guide the model to achieve better performance. When LHS uses random graphs instead of action graphs, the ROUGE scores are even lower than the BART baseline. This indicates that the structural information of the action graph, rather than the node entities is essential to the extraction of to-do items.

Effectiveness of perturbation The perturbation is intended to improve the model's performance in extracting the correct person names. To show

Type	Percentage	ROUGE-2	Ground Truth	Wrong Prediction	Email Text
multiple todos	26%	0	Check out the action.	Try to create a new account.	I'll check out the action. ... I'll also try to create a new account
incorrect action with correct objects	14%	0.36	Send the second item to Lewis.	Get the second item updated for Lewis.	To: Lewis, ... We have sent the second item to you. And we'll update it next week.
correct action with incorrect objects	8%	0.5	Look into the application of creating a new account.	Look into the new account.	You applied to create a new account. We'll look into it.
missing objects	26%	0.4	Get Darshan some mockups asap for intuit status.	Get some mockups asap.	As for intuit status, I'll get some mockups asap.
hallucination	12%	0.4	Keep Dan updated on the meeting.	Keep Darshan† updated on the schedule.	To Dan: .. I will keep you updated.

Table 6: Statistics for 50 samples with lowest ROUGE-2 scores in the validation set. † is a name in the corpus but not mentioned in the email.

its effectiveness, we evaluate its performance on person names independently from other parts of a to-do item by comparing the names in generated to-dos and reference to-dos. Such comparison can be done by extracting the action triplet from the prediction and the reference.

For a to-do item, the names in it correspond to the third part of the corresponding action triplet (the WHAT in WHO-DOING-WHAT), so we extract the mentioned names of a to-do item similarly to the method in Section 3.1. For a pair of action triplets WHO1-DOING1-WHAT1, WHO2-DOING2-WHAT2 extracted from the prediction and the reference, we compute ROUGE-2 score for DOING1, DOING2 and WHAT1, WHAT2 separately. The ROUGE-2 score for WHAT1 and WHAT2 indicates how well the model is able to capture the correct person names. The results are reported in Table 5. When perturbation is applied during training, the ROUGE-2 score for person names increase by 0.03, which indicates that our perturbation trick is effective in alleviating hallucinations about names.

4.6 Challenges and Error Analyses

To examine the challenges our models were faced with, we analyzed the errors made by our method on the test set. We manually examine 50 data points with the least ROUGE-2 scores and put them into different categories of challenges, as shown in Table 6. For each type of challenge we manually picked a sample that shows how the prediction is problematic. The major types include:

- **Multiple todos**: Instead of being a major challenge for models, this is more like a dataset annotation issue. Some data points in the SmartToDo (Mukherjee et al., 2020) dataset actually contain more than one todo items, but the ground-truth has only one item. The annotator and the model selected two different todo items to generate, though both belong to the *actual* ground-truths.
- **Missing objects:** The correct action related to the to-do item is extracted, but the information is not sufficient as the objects of such action are missing.
- **Correct action with incorrect objects:** The correct action has been extracted, but the objects of the action have been assigned to incorrect objects. This kind of error usually happens when the object of the action is not in the same sentence of the action.
- **Incorrect action with correct objects:** The correct objects are extracted, but they are associated with incorrect actions. This kind of error is usually related to multiple sentences or events expressed in the future tense.
- **Extrinsic hallucination:** The incorrect action problem and the incorrect object problem mentioned before belong to *intrinsic hallucinations* which extract information from the email but matches it in a wrong way. While in *extrinsic hallucinations*, the objects of the to-do item are not extracted from the email, but generated from similar entities in the corpus.

5 Conclusion

In this work, we propose a simple yet effective learning to highlight and summarize framework (LHS) to learn to identify salient text and actions from both email text and the constructed action graph, and generate faithful to-do items jointly. Experiments show that the proposed model outperforms the baselines significantly and achieves state-of-the-art performance in both quantitative evaluation and human judgement. We further demonstrated that our LHS framework generalized well to out-of-domain conditions. In the future, we plan to extend the email to-do item generations to other domains such as generating multiple to-do action items from conversation or meeting threads.

Acknowledgement

The authors would like to thank the reviewers for their feedback. This work is funded in part by an Amazon Research Award and a grant from Cisco.

Ethical Discussion

The dataset used in this work is released by Smart-ToDo (Mukherjee et al., 2020), based on the Avocado dataset from the Linguistic Data Consortium, which mainly contains emails from an anonymous defunct technology company referred to as Avocado. The Avocado email dataset entered the public domain via the cooperation and consent of the legal owner of the corpus. Although we do not encounter any major ethical issues in training these models, we anticipate there might be key challenges when it comes to the deployment of email to-do item generation systems, given the highly sensitive and personal nature of emails. We strongly recommend any future use of email to-do generation models only do it in a secure and private manner, and call for a code of AI ethics that lays out how various issues will be handled around deploying these systems for real world uses.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Trung Bui, Matthew Frampton, John Dowding, and Stanley Peters. 2009. Extracting decisions from multi-party dialogue using directed graphical models and semantic similarity. In *Proceedings of the SIGDIAL 2009 Conference*, pages 235–243, London, UK. Association for Computational Linguistics.

Jiaao Chen and Diyi Yang. 2020. Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4106–4118, Online. Association for Computational Linguistics.

Jiaao Chen and Diyi Yang. 2021. Structure-aware abstractive conversation summarization via discourse and action graphs. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1380–1391, Online. Association for Computational Linguistics.

Sihao Chen, Fan Zhang, Kazoo Sone, and D. Roth. 2021. Improving faithfulness in abstractive summarization with contrast candidate generation and selection. In *NAACL*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Xiachong Feng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2020. Incorporating commonsense knowledge into abstractive dialogue summarization via heterogeneous graph networks. *arXiv preprint arXiv:2010.10044*.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 364–372.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Chih-Wen Goo and Yun-Nung Chen. 2018. Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts. *2018 IEEE Spoken Language Technology Workshop (SLT)*.

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268.

Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5094–5107, Online. Association for Computational Linguistics.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020. Semsum: Semantic dependency guided neural abstractive summarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8026–8033.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation.

S. Lahiri, R. MIHALCEA, and Po-Hsiang Lai. 2016. Keyword extraction from emails. *Natural Language Engineering*, 23:1–23.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Manling Li, Lingyu Zhang, Heng Ji, and Richard J. Radke. 2019. Keep meeting summaries on topic: Abstractive multi-modal meeting summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2190–2196, Florence, Italy. Association for Computational Linguistics.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics.

Chu-Cheng Lin, Dongyeop Kang, Michael Gamon, Madian Khabsa, Ahmed Hassan, and Patrick Pantel. 2017. Actionable email intent modeling with reparametrized rnns.

Chunyi Liu, Peng Wang, Jiang Xu, Zang Li, and Jieping Ye. 2019a. Automatic dialogue summary generation for customer service. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD19, page 1957–1965, New York, NY, USA. Association for Computing Machinery.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *Proceddings of the Intertional Conference on Learning Representatinos (ICLR)*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Zhengyuan Liu, Angela Ng, Sheldon Lee, Ai Ti Aw, and Nancy F. Chen. 2019b. Topic-aware pointer-generator networks for summarizing spoken conversations. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Sudipto Mukherjee, Subhabrata Mukherjee, Marcello Hasegawa, Ahmed Hassan Awadallah, and Ryen White. 2020. Smart to-do: Automatic generation of to-do items from emails. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8680–8689, Online. Association for Computational Linguistics.

Smaranda Muresan, Evelyne Tzoukermann, and Judith L. Klavans. 2001. Combining linguistic and machine learning techniques for email summarization. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*.

Gabriel Murray, S. Renals, and J. Carletta. 2005. Extractive summarization of meeting recordings. In *INTERSPEECH*.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the Human Language Technology Con-*

ference of the NAACL, Main Conference, pages 367–374, New York City, USA. Association for Computational Linguistics.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. 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 1747–1759.

A. Nenkova and A. Bagga. 2003. Facilitating email thread access by extractive summary generation. In *RANLP*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Kechen Qin, Lu Wang, and Joseph Kim. 2017. Joint modeling of content and discourse relations in dialogues. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 974–984.

S. Radicati and Q. Hoang. 2011. Email statistics report, 2011-2015.

Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, and Junji Tomita. 2020. Abstractive summarization with combination of pre-trained sequence-to-sequence and saliency models. *arXiv preprint*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674, Melbourne, Australia. Association for Computational Linguistics.

Kaiqiang Song, Chen Li, Xiaoyang Wang, Dong Yu, and Fei Liu. 2020. The ucf podcast summarization system at trec 2020. *arXiv preprint arXiv:2011.04132*.

Peter Turney. 2000. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2:303–336.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405.

Mingfeng Xue, Hang Zhang, and Jiancheng Lv. 2020. *Key Factors of Email Subject Generation*, pages 668–675.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*.

Rui Zhang and Joel Tetreault. 2019. This email could save your life: Introducing the task of email subject line generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 446–456, Florence, Italy. Association for Computational Linguistics.

Zhou Zhao, Haojie Pan, Changjie Fan, Yan Liu, Linlin Li, Min Yang, and Deng Cai. 2019. Abstractive meeting summarization via hierarchical adaptive segmental network learning. In *The World Wide Web Conference*, WWW '19, page 3455–3461, New York, NY, USA. Association for Computing Machinery.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020a. Boosting factual correctness of abstractive summarization with knowledge graph. *arXiv preprint*.

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020b. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

A Prediction Samples

Email Text	Ground Truth	Prediction
Divarkar: Are we able to proceed with development without submitting applications? Attached is a copy of the updated Palm DDS. The customer has approved it. I will let you know if any changes are needed.	Let Divakar know if any changes in palm DDS are needed	Let Divakar know if any changes are needed in palm DDS .
While the customer was testing the Quicken Palm app using the stored cookies, he discovered a few places wrong. This is excellent. Actually I prepared a check list. Certainly we will test in all aspects and try to finish by Friday afternoon PST. But we better not to commit the time to customer I believe. Please let me know if someone is testing on real devices. I will directly co-ordinate with them.	Test Palm app in all aspects and try to finish by Friday afternoon PST .	Test in all aspects of real devices and try to finish by Friday afternoon PST.
Due to a variety of circumstances, we did not migrate employees to the UBSWE environment tonight (Monday, February 4, 2002). We will communicate further details of the migration plan on Tuesday, February 5, 2002. Thank you for your cooperation in advance, please contact the Resolution Center at x 3-1411 should you have any questions. UBSW Energy Global Infrastructure	Communicate further details of the migration to the UBSWE environment	Communicate further details of the UBSW environment.
May, I need a number where you can be reached on Friday evening. I will be getting the schedule after work on Friday and will call and let you know when and where our games will be. D	Let May know about when and where games will be	Let May know about the schedule for the game

Table 7: Four samples randomly picked from the predictions of the model, where the first two are from SmartToDo (the same corpus as the training set), and the second two are from Enron Email (a different email corpus).

B Implementation Details

Following the setting of SmartToDo (Mukherjee et al., 2020), we used 1000 email instances each for validation and testing. We initialize the token-level encoder and the decoder layer that attends to input tokens with pre-trained BART-base (Lewis et al., 2020) parameters and randomly initialize the newly added components. For the components initialized with BART, we employ an initial learning rate of 5e-5. For the newly added components, we employ an initial learning rate of 5e-4. We set the number of warm-up steps to 100. We set the α, the coefficient of $L_{classification}$ to be 1.7 and β, the coefficient of $L_{generation}$ to be 0.3. We fine-tune each model for 7 epochs on the training set.