Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Special Track on AI in FinTech

A Unified Model for Financial Event Classification, Detection and Summarization

Quanzhi Li and **Qiong Zhang**
Alibaba Group
{quanzhi.li, qz.zhang}@alibaba-inc.com

Abstract

There is massive amount of news on financial events every day. In this paper, we present a unified model for detecting, classifying and summarizing financial events. This model exploits a multi-task learning approach, in which a pre-trained BERT model is used to encode the news articles, and the encoded information are shared by event type classification, detection and summarization tasks. For event summarization, we use a Transformer structure as the decoder. In addition to the input document encoded by BERT, the decoder also utilizes the predicted event type and cluster information, so that it can focus on the specific aspects of the event when generating summary. Our experiments show that our approach outperforms other methods.

1 Introduction

News can play an important role towards influencing stock market trends and other financial related activities. Financial professionals and investors have shown great concerns in the financial events. In many financial news analytic applications, there are three main tasks: classifying them in appropriate event types, clustering (detecting) the documents talking about the same or relevant events into the same event cluster, and automatically generating a summary for each event cluster. Traditionally, this is done in a pipeline approach, and each task has its own independent model, despite that there is inter-dependence among these tasks, and some textual and semantic information can be shared by them.

Multi-task learning has been applied in many NLP tasks, and has shown its ability to improve the performance of these tasks. This paper presents a joint learning model for classifying, clustering and summarizing financial events. It has the following features: 1. We fine-tune a pre-trained BERT model [Devlin et al., 2019] to generate document representation, which is shared by all the three tasks. This shared model will extract the common information and patterns among these three tasks, and the pre-trained BERT model lets us exploit the grammar and semantic information of tokens learned from large amount of text data. 2. The event type information of a document can help clustering model by providing additional information. In our model, the event type hidden state is exploited by the clustering model to do better clustering. 3. The summarization component exploits both the event type information and the cluster information by feeding them into the decoding process, so that it can generate different summary styles for different event types, and make the generation model focus more on topic-specific aspects. 4. Our summarization model uses BERT as encoder and Transformer [Vaswani et al., 2017] as decoder. Because the BERT encoder is pre-trained and the Transformer decoder is not, in order to smoothly integrate these two parts together for better generation performance, we employ two separate optimizers for these two components during the training process.

2 Related Work

The task of event detection has been proposed in the Topic Detection and Tracking program [Allan, 2002]. The objective is to discover new or previously unidentified events. Online clustering-based approaches are popular on detecting open-domain events. Aggarwal and Subbian [2012] proposed a stream-based clustering algorithm on each incoming message. Petrovic et al. [2010] and Wurzer et al. [2015] used a Locality Sensitive Hashing (LSH) to detect and cluster events from high-volume streams in constant time and space. Li et al. [2017a, 2018] extract semantic terms from documents and use them to do event clustering. Multi-task learning has been used in various NLP tasks [Collobert and Weston, 2008; Lan et al., 2017; Wang and Zhang 2017; Li et al., 2019].

Sequence to sequence (seq2seq) learning has been used in a variety of language generation applications. It has attracted much attention in recent years due to the advance of deep learning. Our summarization model also belongs to this widely used seq2seq paradigm [Sutskever et al., 2014]. Rush et al. [2015] and Nallapati et al. [2016] were among the first to use the neural encoder decoder structure in text summarization. See et al. [2017] enhance this model with a pointer generator network which allows it to copy words from the source text. Paulus et al. [2018] present a deep reinforced model for abstractive summarization which handles the coverage problem with an intra-attention mechanism. Celikyilmaz et al. [2018] propose an abstractive system where multiple agents represent the document together with a hierarchical attention mechanism for decoding.

Pre-training has been widely used in natural language processing (NLP) tasks to learn better language representation, and several new pre-trained models have been published recently, such as BERT [Devlin et al., 2019], XLNet [Yang et al., 2019a], RoBERTa [Liu et al., 2019], ALBERT [Lan et al., 2019], ELMO [Peters, et al., 2018], etc. The pre-training on large amount of unlabeled data and fine-tuning with small scale labeled data are helpful for many tasks, and it is also used in the encoder part of our model in this work. Devlin et al. [2019] proposed BERT based on masked language modeling and next sentence prediction, and achieved state-of-the-art results on multiple NLP tasks. There are also some works on pre-training the encoder-decoder model for language generation [Rothe et al., 2019; Edunov et al., 2019; Liu and Lapata, 2019]. The main difference between our generation model and others are that our model uses pre-trained BERT model in the encoder side and uses a non-pre-trained Transformer on the decoder side, and we fine-tune the encoder and train the decoder using two separate optimizers.

3 The Unified Model

Figure 1 shows the high-level model structure. The incoming document is encoded by a pre-trained BERT model, and this encoder is shared by the three tasks. The event type hidden state from the event type prediction model is fed to both the clustering model and the decoder of the summarization model. The cluster hidden state vector is also exploited by the summarization decoder. The right side of the figure is the decoder part, which is based on the Transformer architecture, consisting of 6 layers of decoders. The event type state vector, cluster state vector, and the shared document representation are fed to each of the decoder layers, and used by the multi-head attention layer described later. The event type state vector and cluster state vector are also used at the final softmax layer of the Transformer decoder, in order to add more context to help the decoder choose the correct token.

Shared document representation. We use BERT to encode an incoming document. BERT has been used to fine-tune various NLP tasks, but its application to text generation is not straightforward, since it is trained to predict single word and next sentence, not generating text sequence. This is why our model uses BERT to encode the document, but utilizes Transformer on the decoding side to generate text sequence. Due to the limitation of the input text length of the pre-trained BERT, we take the first 510 tokens from the incoming document and feed it to the BERT model. We observe that, for most news articles and financial announcements, the first 510 tokens are enough to cover their main content. Our preliminary experiment also shows that using the first 510 tokens as input performs better than selecting the same number of important sentences from the document as input.

3.1 Event Type Prediction

We use the shared document representation to predict the event type for the input document. We take the final hidden state of the first token [CLS] as the document representation, then it is fed into a fully connected layer, and finally we get an event type hidden state vector H_t:

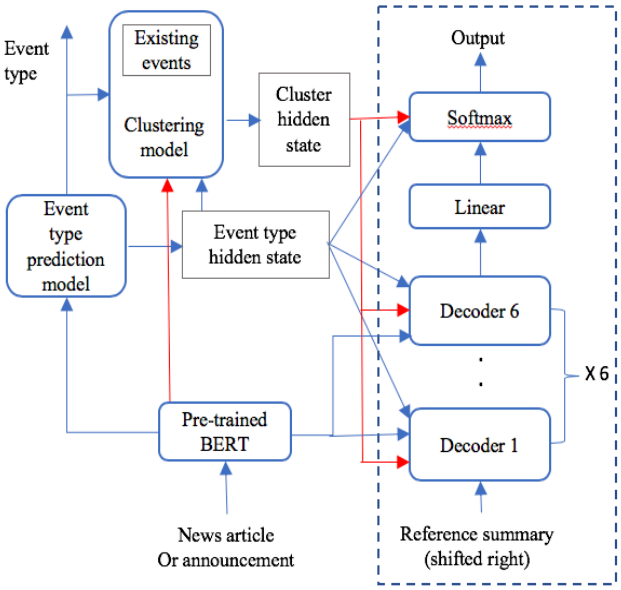

Figure 1. The high-level unified model structure

$$H_t = \sigma(W_t h + b_t) \quad (1)$$

where W_t is the event prediction parameter matrix, b_t is the bias, and h is the [CLS] vector. A softmax operation is then applied to H_t to predict the probability of event type. The pre-trained BERT model is fine-tuned using training data, by minimizing the cross-entropy loss between $\hat{y}$ and y as follow:

$$l(\hat{y}, y) = -\sum_{k=1}^{|Y|} y_k \log(\hat{y}, k) \quad (2)$$

3.2 Event Clustering

For an incoming document, we calculate a similarity score with every document in the existing events. A threshold is used to decide if the incoming document should form a new cluster by itself or it should be assigned to an existing cluster. This value is learned from the training data. If the similarity score is greater than the threshold, then the cluster with the most similar document will be the cluster the incoming document should be assigned to. To compute the similarity score, for each document in the existing clusters, we need two vectors, one is the document's representation vector, which is h in Equation 1, and the other one is the average value of the document representation vectors in this cluster. The average document representation is to represent the cluster. For a real-time system with large number of documents to process, the choice of comparing the incoming document with every document in the existing clusters may have issue of computation time. Another choice is to have one cluster vector, e.g. the average document representation of the cluster, to represent the whole cluster, and the incoming document will be compared to each cluster only once. The second choice may be a better option for a real-time application with large data to handle. In the experiment section, we did an ablation test to see the performance difference between these two choices.

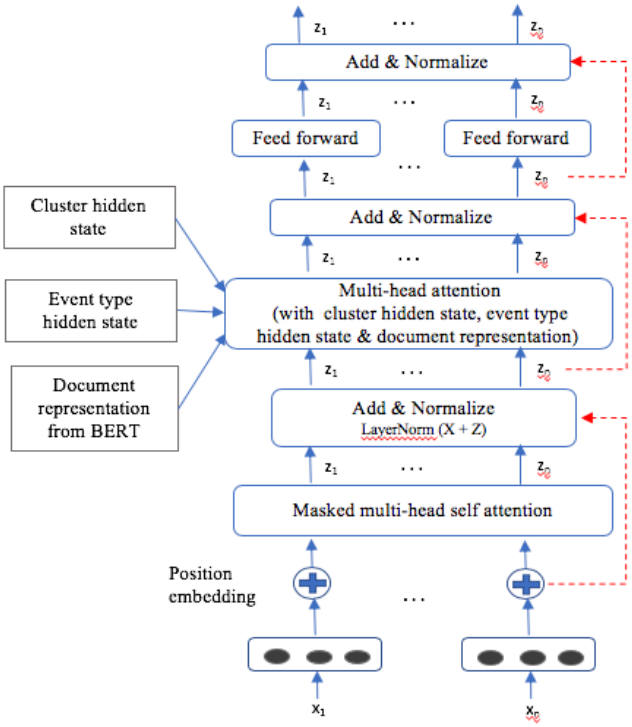

Figure 2. Detailed structure of a decoder layer. This diagram shows the first decoder layer

For a given document in a cluster C, we calculate the cluster hidden state vector H_c as follow:

$$H_c = \sigma(W_c(h + H_t + H_d + H_a) + b_c) \quad (3)$$

Where h is the final hidden state of [CLS] token, H_t is the event type hidden state from Equation 1, H_d is the document representation vector for a document in the compared cluster, H_a is the average of all document vectors of this cluster, and + denotes concatenation of vectors. Then a softmax operation is applied to get the similarity probability score:

$$S = softmax(WH_c + b) \quad (4)$$

Where W and b are model parameters. Based on this score we can find the most similar document, and the cluster of this document is the one where the new document will be assigned to. This approach has been used by previous studies for document clustering [Katiyar and Cardie, 2016; Wang and Zhang 2018].

Cluster merging. It is possible that documents talking about the same event may be placed into different clusters, and gradually we may have cluster segmentation problem. In a real system, we may need a cluster merging process that tries to handle this type of issue. Another purpose of cluster merging is to group related events together. For example, we can group clusters talking about different development stages of an event together into one cluster. The merging process runs periodically to check if we need to merge two clusters together. The merging algorithm is similar to the clustering algorithm.

3.3 Event Summarization

The right part of Figure 1 shows the decoding part of the summarization component. There is a stack of six decoders in the decoding side. Figure 2 presents the detailed structure of a decoder, using the first decoder layer as an example. There are two multi-head attention layers in this decoder, one is a masked multi-head self-attention, and the other one is the decoder attended on the three types of contexts, i.e., the cluster hidden state, the event type hidden state and the document representation vector for the incoming document. Figure 1 has shown where these three vectors come from. After each attention layer and the feedforward layer, there is a Add&Normalize layer.

In our model, a summary is generated for each incoming document. To get a summary for the whole cluster, we apply a summary aggregation process. It works as follow: for the first document in a cluster, its summary will be the summary of this cluster. When we have more than one document in a cluster, we use the approach used in [Erkan and Radev, 2004; Hong and Nenkova, 2014; Yasunaga et al., 2017]. It is basically to choose the sentences that share common information in the document summary set, and at the same time to avoid redundancy. We describe the multi-head attention mechanism in more details below.

Multi-head attention. In the multi-head attention structure, attention is computed not once but multiple times, in parallel and independently. The outputs are concatenated and linearly transformed, as shown in Figure 3. In this figure, m is the number of heads. Figure 4 shows how one head of the scaled do-product attention is computed, which can be expressed as follow:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5)$$

where Q is a matrix that contains the query (vector representation of one word in the sequence), K are all the keys (vector representations of all the words in the sequence), and V are the values. For the first multi-head attention layer in each decoder, V consists of the same word sequence as Q. However, for the second attention module in each decoder, it considers the decoder sequence, the fact embedding, charge embedding and law article embedding, and therefore, here V is different from the sequence represented by Q.

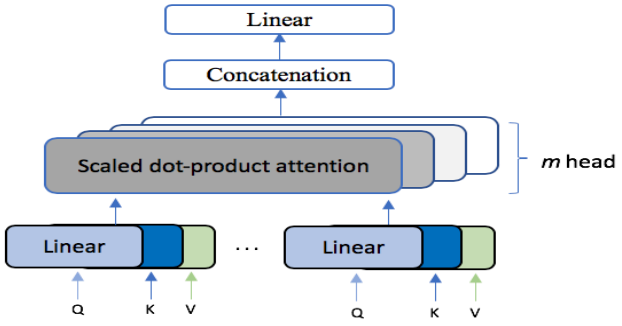

Figure 3. Multi-head attention layer

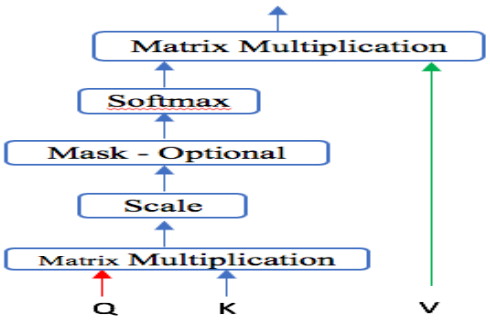

Figure 4. Scaled dot-product attention

3.4 Model Training and Inference

In our model, the document encoder is based on a pre-trained BERT model, and the decoder component is not pre-trained. It is obvious that there is a mismatch between these two components, because one is pre-trained and the other one needs to be completely trained. This may make the training process unstable, e.g. one component is underfitted and the other one is overfitted. One way to handle this is to use two different optimizers for the two components. In our model implementation, two Adam optimizers are used. Each has its own learning rates and warmup steps. These values will be set so that the pre-trained BERT model should be fine-tuned with a smaller learning rate and decay, when the decoder becomes stable. This is to make sure that the BERT model to be fine-tuned with more accurate gradients. The learning rate update is illustrated by the following equation:

$$learnRate = learnRate * min\ (s^{-0.5},\ s * w^{-1.5}) \quad (6)$$

where s is the step and w is the warmup value, similarly to [Vaswani et al., 2017; Liu and Lapata 2019]. For inference of summarization, we use beam search, whose size is set to 4, to find the best sequence. The generated word sequences will be ranked and the one with the largest value will be chosen. The loss of the joint learning model is the sum of loss of the three tasks. There are previous studies on pre-training encoder-decoder model for language generation [Rothe et al., 2019; Edunov et al., 2019; Liu and Lapata, 2019], and some of them also use different optimizers for different components.

4 Experiments and Results

4.1 Data Set and Evaluation Metrics

Data Set

Our data set consists of financial news and announcements in English. These documents belong to seven event categories, shown in Table 1. There are 1600 events, and this table also shows the event percentage for each category. To ensure the quality of the data set, six annotators were trained on analyzing financial events. To help the annotators, we first process these documents by a program mainly based on some rules to classify these documents into appropriate event types and then cluster them. The annotators then work on these intermediate results for event type labeling and clustering. For generating manual summary, they were asked to write *representative* and *informative* summaries for these events. The data set was split into training, validation and evaluation parts, using a 70/10/20 split.

Event type	Description	Events (%)
Merger and acquisition	announcements, forecasts, and cancellations of a merger/acquisition.	10.0
Management change	resignation & appointment of board directors and executives	17.3
Share change	share buyback, stock split, reverse stock split, etc.	20.2
Dividend	dividend announcements, forecasts, payments, stable yields, raises, and reductions	13.7
Debt	debt announcements, forecasts, increases, reductions, etc.	12.5
New product / market	announcement or analysis of new product, technology, etc.	16.2
Other	other news or announcements	10.1

Table 1. Event types and description.

Evaluation Metrics

For event type classification, we use macro F1 measure, which is calculated from precision and recall. For event clustering, we use two quality metrics: Normalized Mutual Information (NMI) [Manning et al., 2008] and B-cubed [Amigo et al, 2008]. They have been used in previous studies. We chose them because both metrics balance the desired clustering properties: to maximize the homogeneity of events within each cluster, and to minimize the number of clusters that documents of each event spread across. Cluster level precision and recall can also be used as quality metrics, but they cannot measure the cohesiveness within a cluster.

NMI. NMI measures how much information is shared between actual "ground truth" events, each with an associated document set, and the clustering assignment. Specifically, for a set of clusters $C = (C_1, C_2, .., C_j)$ and events $E = (E_1, E_2, ..., E_k)$, where each C_j and E_k is a set of documents, and n is the total number of documents, NMI is defined as:

$$NMI\ (C, E) = \frac{I(C,E)}{(H(C)+ H(E))/2} \quad (7)$$

Where

$$I(C,E) = \sum_j \sum_k \frac{|e_k \cap c_j|}{n} log \frac{n * |e_k \cap c_j|}{|e_k| * |c_j|}, \quad (8)$$

$$H(C) = -\sum_j \frac{|c_j|}{n} log \frac{|c_j|}{n} \quad (9)$$

And,

$$H(E) = -\sum_k \frac{|e_k|}{n} log \frac{|e_k|}{n} \quad (10)$$

B-Cubed. It estimates the precision and recall associated with each document in the dataset individually, and then uses

the average precision P_b and average recall R_b values for the dataset to compute B-Cubed:

$$B\text{-}Cubed = \frac{2 * P_b * R_b}{P_b + R_b} \quad (11)$$

For each document, precision is defined as the proportion of items in the document's cluster corresponding to the same event, and recall is the proportion of documents corresponding to the same event.

For summarization, we use three ROUGE scores [Lin and Hovy, 2003] as the evaluation metrics. ROUGE basically measures the overlap of N-grams between the system and reference texts. However, the original ROUGE measure does not tell you much as a metric. To get a good quantitative value, in the context of ROUGE, we compute precision and recall using the overlap, and then report the F1-measure of ROUGE. In this study, we use ROUGE-1, which is based on unigram overlap, ROUGE-2, which is based on bigram overlap, and also ROUGE-L, which measures the longest common subsequence of reference and generated texts, to compute F1.

4.2 Compared Methods and Experiment Setting

Compared Methods

For event type classification. Our approach is compared to the following methods: HAN [Yang et al., 2016] - Hierarchical attention networks for document classification; ULMFiT - the language model fine-tuning method [Howard and Ruder, 2018]. The BERT-base model – adding a text classification layer on top of it.

For clustering. UMass – an old but still very effective and popular event detection algorithm based on tf.idf and cosine similarity by Allan et al [Allan et al., 2000a, 200b]. LSH [Wurzer et al., 2015] - it uses Locality Sensitive Hashing (LSH) to detect and track events on unbounded high volume data streams in constant time and space. SemEntity – an approach based on calculating the similarity between two documents using extracted entities and semantic terms extracted [Li et al., 2017a; Li et al., 2018]. BERT-sim – use the document representation generated by BERT, and calculate similarity between two documents using this representation.

For summarization. Bi-LSTM-attention - this approach is based on seq2seq model with attention, and both the encoder and decoder use a bidirectional-LSTM model [Sutskever et al., 2014; Vaswani et al., 2017; Ye et al., 2018]. Attention mechanism can catch the important input information for the current output sequence. DCA - Celikyilmaz et al. [2018] use multiple encoders to represent the document together with a hierarchical attention mechanism for decoding. Their proposed Deep Communicating Agents (DCA) model is trained end-to-end with reinforcement learning.

Experimental Settings and Training

Like previous studies, we use the validation data set to tune our model and hyper-parameters. Training process was terminated if the model performance is not improved for successive 10 times. We use the BERT-base-uncase model as the encoder for encoding documents. This model uses 12 encoder layers, and the embedding size is 768 for the input token, the position embedding and the text segmentation embedding. The multi-head attention has12 heads, drop out is 0.1, and L2 decay rate is 0.01. The summarization decoder side has six layers, as illustrated in Figure 1. The input is shifted one token position to the right, utilizing a teacher forcing learning approach. The other hype-parameters on the decoder side use the default values of the original Transformer architecture.

As described before, in our model, the BERT encoder and the decoder use two different optimizers. Their learning rates are different. We use two Adam optimizers with $\beta 1 = 0.9$ and $\beta 2 = 0.999$ for the encoder and the decoder, respectively, but they have different learning rates and warmup-steps. In Equation 6, we set $learnRate = 2e-3$ and $warmUp = 30{,}000$ for the BERT encoder, and we set $learnRate = 0.05$ and $warmUp = 15{,}000$ for the decoder.

4.3 Evaluation Result and Analysis

Experiment result. Table 2, 3 and 4 present the comparison results. R-1, R-2, R-L are the F1 values of ROUGE-1, ROUGE-2 and ROUGE-L, respectively. The result shows that our proposed approach performs better than other methods. To verify if the performance improvement is statistically significant, we conducted *t-test* between our model and others. The *t-test* results show that the performance improvements are statistically significant at the level of $p=0.05$, for the evaluation metrics of event clustering and summarization tasks. It also outperforms the two baseline methods on event type classification, but the difference is not statistically significant. The result also shows that for event clustering UMass is better than LSH, which is similar to the results reported by [Petrovic et al., 2012; Wurzer et al., 2015], in terms of the quality of clusters. The reason is that the focus of LSH is speed, not clustering accuracy.

Method	F1
HAN	92.5
ULMFit	93.3
BERT	94.8
Our approach	95.5

Table 2. Event type classification result

Method	NMI	B-Cubed
UMass	0.665	0.310
LSH	0.613	0.263
SemEntity	0.708	0.391
BERT-sim	0.717	0.397
Our approach	0.742	0.414

Table 3. Event clustering result

Method	R-1	R-2	R-L
Bi-LSTM-attention	52.8	34.0	49.2
DCA	54.1	35.5	50.9
Our approach	56.5	37.7	53.3

Table 4. Event summarization result

Method	NMI	B-Cubed
Our approach	0.742	0.414
- No event type hidden state	0.738	0.411
- Use only the average document state of an event for similarity calculation	0.730	0.402
- Use only the closest document state of an event for similarity calculation	0.734	0.406

Table 5. Ablation test result on clustering

Method	R-1	R-2	R-L
Our approach	56.5	37.7	53.3
- No event type info	55.2	36.8	52.2
- No event type info in the final softmax output layer	55.9	37.3	52.7
- No cluster info	54.9	36.5	52.1
- No cluster info in the final softmax output layer	55.3	36.7	52.4

Table 6. Ablation test result on summarization

Ablation Test

The experiment results have shown that the joint leaning approach, where the three tasks share common document representation generated by the pre-trained BERT model, do improve the performance for all three tasks. In our proposed model, as shown in Figure 1, the hidden state of event type is also fed to the clustering and summarization components, and the hidden state of the cluster where the document is assigned is also exploited by the summarization component. To see how these features affect the performance of clustering and summarization, we conducted ablation tests. The result is presented in Table 5 and 6.

For clustering task, we can see that excluding the hidden state vector of event type from the similarity calculation Equation 3 slightly affected the clustering performance (e.g. NMI changed from 0.742 to 0.738). One reason the performance did not go down much is that we only compare the document to the event clusters in the same evet type as the document, which means the event type information is already considered, and so removing this hidden feature vector from the similarity calculation will not affect the clustering performance much. Table 5 also shows that when calculating similarity with existing clusters, the performance will decrease if we use only the average document state of all documents in an event or use only the most similar document state of an event in Equation 3.

For summarization task, Table 6 shows that excluding either the event type information or the cluster information will drag down the performance. And the cluster information will affect the performance a little more. One reason is that the cluster information will give the decoder a more complete picture about the cluster. The Event type information can provide more information about the specific event type. For example, the event arguments we want to mention in the summary for a merger and acquisition event will be very different from a management change event. The arguments of a merger and acquisition event usually involve acquirer, target, time and the amount of money, while management change event will involve person name, title and time. Therefore, event type information can help the generation model know what aspects should be focused on for a specific event type. This ablation test also shows that adding event type or cluster information to the final softmax layer will improve the generation performance, since they provide more context for decoder to choose the right token.

5 Conclusion

We present a joint learning model for financial event classifying, clustering and summarizing. This uses pre-trained BERT to encode the incoming document, and the encoded information are then shared by the event type classification, detection and summarization components. For event summarization, we use a Transformer structure as the decoder. In addition to the input document encoded by BERT, the decoder also considers the predicted event type and cluster information, so that it can focus on the specific aspects of the event type when generating summary. The experiment results show that our approach outperforms other compared methods.

References

[Gottlob et al., 2002] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, May 2002.

[Gottlob, 1992] Georg Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, June 1992.

[Aggarwal and Subbian, 2012] Charu C Aggarwal and Karthik Subbian. Event detection in social streams. In SDM, volume 12, pages 624–635. SIAM 2012.

[Allan et al., 2000a] James Allan, Victor Lavrenko, and Hubert Jin, First story detection in TDT is hard. CIKM 2000

[Allan et al., 2000b] James Allan, Victor Lavrenko, Daniella Malin, and Russell Swan, Detections, bounds, and timelines: UMass and TDT–3. TDT–3 2000.

[Allan, 2002] James Allan, editor. Topic Detection and Tracking: Event-based Information Organization. Kluwer Academic Publishers, 2002.

[Amigo et al., 2008] Enrique Amigó, Julio Gonzalo, Javier Artiles and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information Retrieval, 2008.

[Celikyilmaz et al., 2018] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. Deep communicating agents for abstractive summarization. NAACL 2018

[Collobert et al., 2011] Ronan Collobert, Jason Weston, L´eon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. JMLR, 12:2493–2537, 2011.

[Devlin et al., 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep

bidirectional transformers for language understanding. NAACL 2019

[Edunov et al., 2019] Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. NAACL 2019

[Erkan and Radev, 2004] Gunes Erkan, Dragomir R. Radev. Lexrank: graph-based centrality as salience in text summarization, JAIR 2014

[Hong and Nenkova 2004] Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multidocument summarization. EACL 2004

[Howard and Ruder, 2018] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 .

[Katiyar and Cardie, 2016] Arzoo Katiyar and Claire Cardie. Investigating lstms for joint extraction of opinion entities and relations. ACL 2016.

[Lan et al, 2019] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut, ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, https://arxiv.org/abs/1909.11942, 2019

[Li et al., 2017a] Quanzhi Li, Armineh Nourbakhsh, Sameena Shah, Xiaomo Liu, Real-Time Novel Event Detection in Twitter, IEEE ICDE 2017

[Li et al., 2018] Quanzhi Li, Armineh Nourbakhsh, Sameena Shah, Systems and methods for event detection and clustering, US Patent App. 15/800,876, 2018.

[Li et al., 2017b] Quanzhi Li, Sameena Shah, Xiaomo Liu, Armineh Nourbakhsh, Data sets: Word embeddings learned from tweets and general data, The 11th International AAAI Conference on Web and Social Media (ICWSM-2017)

[Li et al., 2019] Quanzhi Li, Qiong Zhang, Luo Si, Rumor Detection by Exploiting User Credibility Information, Attention and Multi-task Learning, ACL 2019

[Lin and Hovy, 2003] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. NAACL 2003

[Liu and Lapata, 2019] Yang Liu and Mirella Lapata, Text Summarization with Pretrained Encoders, EMNLP 2019

[Liu et al., 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al., RoBERTa: A Robustly Optimized BERT Pretraining Approach, https://arxiv.org/abs/1907.11692

[Liu et al., 2017] Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, et al., Reuters Tracer: Toward Automated News Production Using Large Scale Social Media Data, IEEE BigData 2017

[Mannin g et al., 2008] Christopher D. Manning. Prabhakar Raghavan. Hinrich Schütze. Introduction to Information Retrieval. Cambridge Univ. Press, 2008

[Mueller and Thyagarajan, 2016] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. AAAI 2016.

[Nallapati et al., 2016] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, Bing Xiang. Abstractiv text summarization using sequence-tosequence RNNs and beyond. CoNLL 2016

[Peters, et al., 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer, Deep contextualized word representations, NAACL 2018

[Petrovic et al., 2010] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to Twitter. HLT-ACL 2010

[Rothe et al., 2019] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. arXiv preprint arXiv:1907.12461, 2019

[Rush et al., 2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. EMNLP 2015

[See et al., 2017] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer generator networks. ACL 2017

[Strehl et al., 2002] Alexander Strehl, Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. JMLR, 3:583 - 617, 2002.

[Sutskever et al., 2014] lya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. NIPS 2014

[Wang and Zhang, 2017] Zhongqing Wang and Yue Zhang, A Neural Model for Joint Event Detection and Summarization, IJCAI 2017

[Wurzer et al. , 2015] DominikWurzer, Victor Lavrenko, and Miles Osborne. Twitter-scale new event detection via kterm hashing. EMNLP 2015.

[Yang et al., 2019a] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le, XLNet: Generalized Autoregressive Pretraining for Language Understanding, NIPS 2019

[Yang et al., 2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. NAACL 2016

[Yasunaga et al., 2017] Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, et al. Graph-based Neural Multi-Document Summarization, CoNLL 2017Conference Name:ACM Woodstock conference

[Ye et al., 2018] Hai Ye, Xin Jiang, Zhunchen Luo, Wenhan Chao, Interpretable Charge Predictions for Criminal Cases: Learning to Generate Court Views from Fact Descriptions, NAACL 2018