IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 October 2022, accepted 4 November 2022, date of publication 11 November 2022, date of current version 17 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3221404

RESEARCH ARTICLE

Extractive Summarization of Call Transcripts

PRATIK K. BISWAS[1] AND ALEKSANDR IAKUBOVICH[2]
[1]Artificial Intelligence and Data, Global Network and Technology (GNT), Verizon Communications, Basking Ridge, NJ 07920, USA
[2]Core Engineering and Operations, Global Network and Technology (GNT), Verizon Communications, Richardson, TX 75081, USA
Corresponding author: Pratik K. Biswas (pratik.biswas@verizonwireless.com)

ABSTRACT Automatic text summarization is one of the most challenging and interesting problems in natural language processing (NLP). Text summarization is the process of extracting the most important information from the text and presenting it concisely in fewer sentences. Call transcript involves textual description of a phone conversation between a customer (caller) and agent(s) (customer representatives). Call transcripts pose unique challenges that are not adequately addressed by most open-source automatic text summarizers, which are developed to summarize continuous texts such as articles and stories. This paper presents an indigenously developed method that combines topic modeling and sentence selection with punctuation restoration in condensing ill-punctuated or un-punctuated call transcripts to produce more readable summaries. This unique combination is what distinguishes the proposed summarizer from other text summarizers. Extensive testing, evaluation and comparisons, with an open-source, state-of-the-art extractive summarizer using three different pre-trained language models, have demonstrated the efficacy of this summarizer for call transcript summarization. The summaries generated by the proposed summarizer are shown to be more compelling and useful based on multiple criteria.

INDEX TERMS Extractive summarization, topic models, transformers, embedding, punctuation restoration.

I. INTRODUCTION

In recent years, there is an abundance of multi-sourced information available for public consumption, fueled by the growth of the Internet. In many cases, this volume of readily available text requires effective summarization for different purposes. It is very difficult for humans to manually summarize large quantities of text. Hence, automatic text summarization has become a desirable tool in today's information age. It produces concise, fluent and readable summaries from larger bodies of text, while preserving the original information content and meaning. Such summarization can be very useful when applied to various domains such as news articles, emails, call transcripts, medical history, and mobile text messages. Many such summarizers are available online on the Internet, including Microsoft News2 and Google1 for news articles [1], MEAD and SWESUM for biomedical information [2], and WikiSummarizer for Wikipedia articles.

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

Numerous approaches have been developed for automatic text summarization, which can be broadly classified into two groups: extractive and abstractive summarization. Extractive summarization extracts important sentences from the original text and reproduces them verbatim in the summary, whereas abstractive summarization generates new sentences.

Call transcripts are written texts originally presented in a different medium; thus, call transcription is defined as the process of converting a voice or video call audio track into written words through speech-to-text conversion, to be stored as plain text in a conversational language. In this paper, however, we will confine ourselves to textual descriptions of audio recordings of voice calls between the customer (caller) and agent(s) (customer representatives) of a phone company. Automatic summarization of call transcripts, in our consideration, pose certain unique challenges, as follows: 1) they are not continuous texts but include conversation between customers and agents, 2) they are often very long and are embedded with "small talks" and can include a large number of sentences that are irrelevant and even meaningless, 3) they include several ill-formed, grammatically incorrect

sentences, 4) they are either un-punctuated or are improperly punctuated based on *pauses* in the conversation as perceived by annotators and so are often unreadable, and 5) existing open-source summarization tools don't perform too well with call transcripts. *Hence, a new domain-specific text summarizer* is *required*.

In this paper, we present a novel extractive summarization technique that combines *channel separation* (separation into customer and agent transcripts), *topic modeling, and sentence selection* with *punctuation restoration* to produce properly punctuated, fixed-length and readable customer and agent summaries from the original call transcripts, which can adequately summarize customer concerns and agent resolutions.

II. RELATED WORK

Related research can be broadly grouped into two categories: 1) extractive summarization and 2) abstractive summarization. Research in the first category is most relevant to our work.

Radef et al. [3] defined summary as ''a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that.'' Automatic text summarization gained attention as early as the 1950s. Different methods and extensive surveys of automatic text summarization have been provided in [1], [4], [5], [6], [7], [8], [9], [10], [11], [12], and [13].

Luhn et al. [14] introduced a method for extracting salient sentences from text using features such as word and phrase frequencies. They proposed weighting the sentences of a document as a function of high-frequency words, ignoring very high-frequency common words. Edmundson et al. [15] described a paradigm based on key phrases where they used four different methods to determine the sentence weight. Kupiec et al. [16] developed the trainable document summarizer, which performed the sentence-extracting task based on a number of weighting heuristics. Bookstein et al. [17] built clusters of index terms, phrases and other subparts of documents for extractive text summarization. Brandow et al. [18] launched the ANES text extraction system that automatically condensed domain-independent electronic news data. Conroy et al. [19] and Mittendorf et al. [20] used hidden Markov models for text summarization. Chen et al. [21] discussed a sentence-selection-based approach to text summarization, whereas Gong et al. [22] and Wang et al. [23] described how multiple documents could be summarized using topic models. Wu et al. [24] suggested a new text-to-graph task for predicting summarized knowledge graphs from long documents, whereas Franciscus et al. [25] used a belief graph data model that aggregated words in a semantic order to generate short texts. Zhong et al. [26] formulated the extractive summarization task as a semantic text-matching problem, in which a source document and candidate summaries (extracted from the original text) were matched in a semantic space. Neto et al. [27] introduced machine learning approaches to automatic text summarization and Kaikhah [28] discussed how neural networks could be useful for summarizing news articles. Suanmali et al. [29] proposed a fuzzy logic based extractive text summarization to improve the quality of summaries created by the general statistical method. Nallapati et al. [30] presented a recurrent neural network (RNN) based sequence model for extractive summarization of documents. Narayan et al. [31] conceptualized extractive summarization as a sentence-ranking task and proposed a novel training algorithm for optimizing the recall-oriented understudy for gisting evaluation (ROUGE) metric [32] using a reinforcement learning objective. Xu et al. [33] constructed a neural model for single-document summarization based on extraction and syntactic compression. Verma et al. [34] applied a restricted Boltzmann machine to enhance the summaries of factual reports created through extractive summarization. Miller [35] used bidirectional encoder representations from transformer (BERT) model [36] to summarize lecture notes. Liu [37] demonstrated that BERTSUM, a simple variant of BERT for extractive summarization, outperformed the previous best performance on the CNN/Dailymail dataset on ROUGE-L scores. Subsequently, Liu et al. [38] demonstrated a general framework for applying BERT to both extractive and abstractive summarizations. Zhang et al. [39] proposed HIBERT for document encoding, designed a method for pre-training it for document modeling using unlabeled data and then applied their pre-trained HIBERT to document summarization to achieve state-of-the-art performance on both CNN/Dailymail and New York Times datasets. Lemberger et al. [40] reviewed several deep learning architectures for automatic text summarization. *Our approach is semantically similar to [18], [22] and [23], but the differences lie in our contributions, as listed in Section III. In addition, unlike in [14], [15] and [18], our term and sentence selections are based on similarity analysis using pre-trained embedding models*.

Lin et al. [41] surveyed the state-of-the-art in abstractive summarization, while Khan et al. [42] reviewed various abstractive summarization methods. Nallapati et al. [43], Paulus et al. [44], See et al. [45] and Liu et al. [46] employed recurrent neural networks, deep reinforcement learning, pointer-generators and generative adversarial networks for abstractive summarization. More recently, Savelieva et al. [47] used BERTSUM with transfer learning to generate summaries of narrated instructional videos across topics ranging from gardening and cooking to software configuration and sports.

III. MAJOR CONTRIBUTIONS

Our main contributions and advantages are as follows:

1) We integrate topic modeling and embedding based sentence selection with transformer based punctuation restoration for extractive summarization through a novel 10-step sequential method (procedure).
2) Our method splits the original call transcript into customer and agent transcripts using the associated

channel identifiers and then summarizes each transcript separately for more coherent results.
3) Our method restores full punctuation to the summaries of un-punctuated or ill-punctuated call transcripts.
4) We uniquely modify and retrain the BERT transformer model architecture for punctuation restoration by adding a classification layer above the 12 layers of BERT.
5) Our method creates, compares and evaluates the performances of different types of topic models for the transcripts, before selecting the most optimal one for summarization. It also provides the option to specify the topic model type to be used for extractive summarization and allows the summarizer to use different topic model types for customer and agent summaries.
6) We introduce a new metric for measuring the effectiveness of punctuation restoration in the punctuated summaries.
7) We demonstrate that the proposed summarizer can outperform other open-source, popular, state-of-the-art extractive summarizers in summarizing call transcripts.

IV. PRELIMINARIES – CONCEPTS AND TECHNOLOGIES

In this section, we clarify key concepts and terminologies and explain certain technologies, which provide the foundation for our work.

The automatic summarization of text is a well-defined task in the field of *natural language processing (NLP)*. *Automatic text summarization* attempts to convert a larger document into a shorter version while preserving its information content and overall meaning. A good summary should reflect the diverse topics of a document while maintaining minimum redundancy [7]. Next, we look at two different approaches to automatic text summarization: extractive and abstractive.

Extractive summarization methods identify the relevant sections in the original text, select the most important paragraphs, sentences, phrases, etc., and concatenate them into shorter forms. By contrast, *abstractive summarization* methods attempt to convey the most important information from the original text by generating new sentences. In other words, they interpret, examine and analyze the original text using advanced natural language techniques to obtain a better understanding of the content and then describe it through shorter and more focused text, comprising of new sentences. Purely extractive summaries often yield better results than automatic abstractive summaries [48]. This is because abstractive summarization methods cope with problems such as semantic representation, inference and natural language generation, which are relatively more difficult than data-driven approaches such as sentence extraction [4]. Most abstractive summarization techniques, specifically those using deep learning, also depend on extractive summarization to extract the summaries for the training samples from which they train to generate new text. In this study, we focus only on extractive summarization, as it is relevant to our work.

A. EXTRACTIVE SUMMARIZATION

Extractive summarization techniques extract the most important sentences, paragraphs, etc., from the original text. The importance of the sentences is based on their statistical and linguistic features. The input can be either single or multiple documents, or text sources. Extractive summarization consists of three main steps: intermediate representation of the input text, scoring of sentences based on the intermediate representation, and selection of sentences for summary generation. There are two approaches, topic-based and indicator-based, which are used for the intermediate representation of the original text. Topic-representation-based approaches transform input text into constituent topics. These are further grouped into frequency-driven, topic word-based, cluster-based, latent semantic analysis-dependent, and Bayesian topic model-based methods. Indicator-representation-based approaches characterize sentences in the input text through features such as sentence length, position in the document, having certain phrases, etc. These are further grouped into graph-theoretic, fuzzy-logic driven, machine learning-based, and neural network-based methods.

B. LATENT SEMANTIC ANALYSIS

Latent semantic analysis (LSA), also known as *latent semantic indexing (LSI)*, is an *unsupervised* method for extracting a representation of text semantics based on observed words. It attempts to bring out latent relationships within a collection of documents on to a lower-dimensional space. LSA is based on the principle that words that are close in meaning occur in similar pieces of text (the distributional hypothesis). It uses a mathematical technique called *singular value decomposition (SVD)* to identify patterns in the relationships between terms and concepts contained in unstructured texts. This method was introduced by Deerwester et al. in [49]. It has been used for multi-document summarization.

C. BAYESIAN TOPIC MODELS

Topic modeling can be described as a statistical method for finding a group of words (i.e., topic), from a collection of documents, that best represents the information in the collection. Bayesian topic models are *unsupervised probabilistic models* that uncover and represent the topics of documents or source texts [50]. They have gained immense popularity in the recent years. Their advantage in describing and representing topics in detail enables the development of summarizers, which can use them to determine the similarities and differences between documents to be summarized [10].

Many techniques have been used to obtain probabilistic topic models. *Latent dirichlet allocation (LDA)* is a widely used topic modeling technique that represents documents as a random mixture of latent topics, where each topic is a probability distribution of words [51]. It has recently been used for multi-document summarization. The *hierarchical dirichlet process (HDP)* is another topic modeling technique, which is an extension of LDA. It is a nonparametric Bayesian approach that uses a mixed-membership model for unsu-

pervised analysis of grouped data. Unlike LDA (its finite counterpart), HDP infers the number of topics from the data.

D. TRANSFORMERS

Transformers in NLP provide general-purpose architectures for *natural language understanding (NLU)* and *natural language generation (NLG)* with over 32+ pre-trained models. These were first introduced in [52]. Transformers are *Seq2Seq deep learning* models that transform sequential inputs into sequential outputs. However, they are based solely on *attention* mechanisms, dispensing entirely with the *recurrence* and *convolutions* of earlier deep learning architectures. Transformers do not require sequential data to be processed in order, which allows for more parallelization than *recurrent neural networks (RNNs)*, thereby reducing training times [52]. Since their introduction, transformers have become the model of choice for tackling many problems in NLP, replacing older recurrent neural network models such as the *long short-term memory (LSTM)*. Transformer models can train on much larger datasets than before because they can support more parallelization during training. This has resulted in the development of pre-trained systems such as *bidirectional encoder representations from transformers* (*BERT*) [36]. BERT is a bidirectional transformer pre-trained, using a combination of masked language modeling objective and next sentence prediction, on a large corpus comprising the Toronto Book Corpus and Wikipedia, by jointly conditioning on both left and right contexts in all layers. Consequently, a pre-trained BERT model can be fine-tuned with only one additional output layer to create state-of-the-art models for a wide range of NLP tasks [36]. *GPT-2* [53] and *XLNet* [54] are two other recently pre-trained NLP models. GPT-2 was trained on 40 GB dataset called WebText and has approximately 1 billion parameters. It was trained to predict the next word. XLNet is an improved version of BERT that implements *permutation language* modeling in its architecture and randomly predicts the next tokens.

Transformers employ a 12-layered *encoder-decoder* architecture comprising a stack of six encoding layers that processes the input iteratively one layer after another and another stack of six decoding layers that does the same thing to the output of the encoder. The encoders are all identical in structure. Each one is broken down into two sub-layers, namely, *self-attention* and *feed-forward neural network*. The decoder has one more layer between them, which is an attention layer that helps it focus on relevant parts of the input sentence (similar to what attention does in Seq2Seq models). Therefore, when a sentence is passed into a transformer, it is embedded and passed into a stack of encoders. The output from the final encoder is passed to each decoder block in the decoder stack, which then generates the output.

E. EMBEDDINGS

Embeddings are mathematical functions that map ''entities'' to a latent space with complex and meaningful dimensions. Words, sentences or paragraphs can be mapped into a shared latent space so that the meanings of words, sentences or paragraphs can be represented geometrically. Machine learning approaches towards NLP require words to be expressed in vector forms. Word embedding, proposed in [55], is a feature engineering technique in which words are mapped into a vector of real numbers in a pre-defined vector space. It is a learned representation for text in which words with the same meaning have a similar representation. The idea of using a dense-distributed representation for each word is key to this approach. *Word2Vec and GloVe* provide pre-trained word embedding models in a type of *transfer learning*. Embedding techniques initially focused on *words*, but attention soon shifted to other types of textual content, such as *n-grams*, *sentences*, and *documents*. The *universal sentence encoder (USE)* [56] encodes text into high dimensional vectors that can be used for text classification, semantic similarity, clustering, and other natural language tasks. The model is trained and optimized for sentences, phrases, or short paragraphs from a variety of data sources with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The model maps variable length input English text into an output of a 512 dimensional vector.

V. EXTRACTIVE SUMMARIZATION OF CALL TRANSCRIPTS

This section provides a description of an extractive summarization technique that we propose for summarizing call transcripts. This extractive summarization technique uniquely integrates channel (speaker) separation, topic modeling, and similarity based sentence selection with punctuation restoration through a 10-step sequential method (procedure). This procedure is highly parameterized. The following are the ten self-contained steps, each with a brief description.

1) **Call Transcript Channel (Speaker) Separation:** Separate each call transcript into customer and agent transcripts based on its channel (speaker) identifier, by iterating through all transcripts.
2) **Partial Punctuation Restoration:** Preprocess transcripts (customer and agent) to remove existing punctuations and use a transformer-based model to restore punctuations *partially*, that is, restore only *periods* as delimiters, so that sentences can be separated in each; by iterating through all transcripts.
3) **Document Preparation:** Preprocess transcripts and generate *documents* from customer and agent transcripts, by iterating through all transcripts, that is, one document from each transcript, where the document is a list of words obtained through NLP pipeline based preprocessing.
4) **Topic Modeling:** Build and optimize different types of *topic models* using the *vocabularies*, *corpus* and *documents* from all customer and agent transcripts, and then pick the best customer and agent topic models based on their coherence scores.
    a) Build different types of customer and agent topic models, such as LDA, LSI, and HDP, by varying their hyper-parameter (e.g., topic number) values

within pre-specified ranges and evaluate the models using their coherence scores (c_v, u_mass, etc.).
b) Select the optimal (or near) topic models for customer and agent transcripts.

If the ''topic model type'' is provided during the invocation of the procedure, then model optimization is confined to only that topic model type in step 4-a for the best model selection.

5) **Dominant Topic Identification:** Obtain the most *dominant topic(s)* from the aforementioned topic models with the associated keywords for each customer and agent documents in every pair, by iterating through all transcripts.
6) **Significant Term Selection:** Obtain the most *relevant keywords/terms* from each pair of customer and agent transcripts by performing a *term-based similarity analysis* between the keywords of the corresponding dominant topics, using one of the following two approaches, by iterating through all transcripts.
   a) Global Extraction – Extract *terms* from the keywords associated with each pair of dominant topics, which need not necessarily be present in the transcripts (customer and agent) themselves.
   b) Local Extraction – Extract *terms* from (local to) the customer and agent transcripts that are similar to the corresponding dominant topic keywords and are also similar to themselves.

   The ''term extraction method'' can be chosen during invocation of the procedure.
7) **Summary Generation:** Generate fixed-length (user-specified length) customer and agent transcript summaries, by iterating through all pairs of customer and agent transcripts.
   a) Identify the *most unique sentences* in each of customer and agent transcripts in every pair, to reduce the transcripts' lengths if necessary, based on *similarity analysis* among all sentences of the corresponding transcript using sentence embeddings.
   b) Extract a fixed number (user-specified) of most relevant sentences from each of the customer and agent transcripts (reduced) through *sentence-based similarity analysis* between every sentence of the corresponding transcript and the string/document constructed from the most significant terms for that pair of transcripts (step 6), using embeddings.

   The ''desired summary length'' (number of sentences) can be specified during the invocation of the procedure.
8) **Punctuation Restoration:** Remove *existing periods* from each pair of customer and agent summaries, restore *partial* and *full punctuation* using a transformer-based model and post-process to make them more readable, by iterating through all of them.
9) **Summary Tabulation:** Save summaries of all transcripts in a table for future use.
10) **Summarization Efficacy Determination:** Evaluate summaries on *content* (information) and *readability* (punctuation restoration), by iterating through every pair of transcripts and their corresponding summaries.
   a) Summary Evaluation – Evaluate the *goodness* of summarization by comparing customer and agent summaries against original transcripts (or manually generated summaries) to generate average *rouge* and *bleu scores*.
   b) Punctuation Restoration Evaluation – Evaluate the *correctness* of punctuation restoration by matching the number of punctuation symbols (*periods*) between the *extracted* and their *partially punctuated* summaries for both the customer and agent, to generate the average *accuracy scores*.

The *full punctuation restored summaries* from step 8 are the outputs from this method (procedure). The full list of parameters for the proposed procedure includes: Topic Model Type (default: ''None/False''), Number of Topics (default: 5), Number of Dominant Topics (default: 1), Batch Size for Punctuation Restoration (default: 512), Term Extraction Method (default: ''global''), Desired Summary Length (default: 5), Summary Table Name (default: ''summary_results''), Word Similarity Threshold (default: 0.5), and Uniqueness Threshold for Sentence Similarity (default: 0.5). Algorithm1 (Appendix) formalizes this procedure.

Next, we take a deeper look at some key steps of this procedure, discuss their implementations in detail, and provide algorithms where necessary.

A. CHANNEL SEPARATION

Call transcripts include conversations/dialogs between customers and one or more agents; thus the resultant summaries can often get mixed up. The separation of a transcript into customer and agent transcripts can make each summary more coherent. Customer summaries can provide better ideas of the problems, while the agent summaries can provide a better understanding of the causes or solutions.

Call transcripts are generally available as *json-formatted* objects. Hence, channel separation involves extraction of the transcript string from the json-formatted object, channel identification, and decomposition of the transcript into customer and agent transcripts using the associated channel identifiers. If the channel identifiers do not clearly identify the speakers, we can use a *pre-trained BERT transformer* model [36] with a *linear classifier* from the *PyTorch nn* module as an additional layer, on top of BERT's 12 layers, to classify each dialog of the transcript into one of two classes, that is, *customer* and *agent* and then combine each type of dialog to create customer and agent transcripts.

B. DOCUMENT PREPARATION

A *document* is a list of *keywords* extracted from each transcript and used as an input to the topic model. For document

preparation, we built a custom NLP preprocessing pipeline comprising tokenization, punctuation, extended stop-words and small words (length ≤ 4) removal, regular expression matching, lowercasing, contraction mapping, bigrams and trigrams creation, lemmatization, and part-of-speech tagging (and allowable tag selection). This was implemented by combining modules (functionalities) available from four *Python* packages: *re*, *spaCy*, *NLTK*, and *genism*.

C. TOPIC MODEL OPTIMIZATION AND OPTIMAL MODEL SELECTION

If the topic model type (t) is specified at the invocation of the procedure, then we create multiple topic models (TM) of the desired type, for both customer and agent, using the documents (d), corpus (c) and vocabulary (v) from the corresponding transcripts, by varying the hyper-parameter (e.g., topic number or n) values within the pre-defined ranges (e.g., 5-50) by the pre-defined steps; compute (ccs) their coherence scores and identify the topic models ($\text{tm}_{\max}$) and associated hyper-parameter values that produce the best scores. This can be expressed for both the customer and agent using Equation (1).

$$\begin{aligned} \text{tm}_{\max} &= \text{argmax}_{\text{tm}\in\text{TM}}\,(\text{ccs}\,(\text{tm}\,(v, c, d, t, n)) : n \\ &= \text{NumberOfTopics}, .., 50) \end{aligned} \tag{1}$$

Otherwise, by default, we perform the above-mentioned activity for all three different topic model types, namely LDA, LSI, and HDP, in parallel, and identify the topic models and associated hyper-parameter values that produce the best scores among the topic models of all three types. This can be similarly expressed for both the customer and agent using Equation (2).

$$\begin{aligned} \text{tm}_{\max} &= \text{argmax}_{t\in\{\text{LDA,LSI,HDP}\}} \\ &\quad (\text{ccs}(\text{argmax}_{\text{tm}\in\text{TM}}(\text{ccs}(\text{tm}(v, c, d, t, n)) : n \\ &= \text{NumberOfTopics}, .., 50))) \end{aligned} \tag{2}$$

Figure 1 shows the steps involved in this algorithm. For topic modeling, we used the *Python* based *genism* package extensively.

D. PUNCTUATION RESTORATION

Here, we describe the punctuation restoration algorithm, used in steps 2 and 8 of the aforesaid (proposed) procedure in detail.

We used the *BertForMaskedLM* class of the *PyTorch BERT* model (*bert-base-uncased*) [36], [57] for punctuation restoration and added an additional *linear* layer (*PyTorch nn module*) above the 12 BERT layers. The output of the original BERT layers is a vector of the size of all vocabularies. The additional linear layer takes this as input and gives as output one of four classes, that is, ''O'' (Other), ''Comma'', ''Period'', and ''Question'' for each encoded word. We retrained this modified BERT model using *TED transcripts*, consisting of two million words. Different variations of punctuation restoration with the BERT model have been presented earlier;

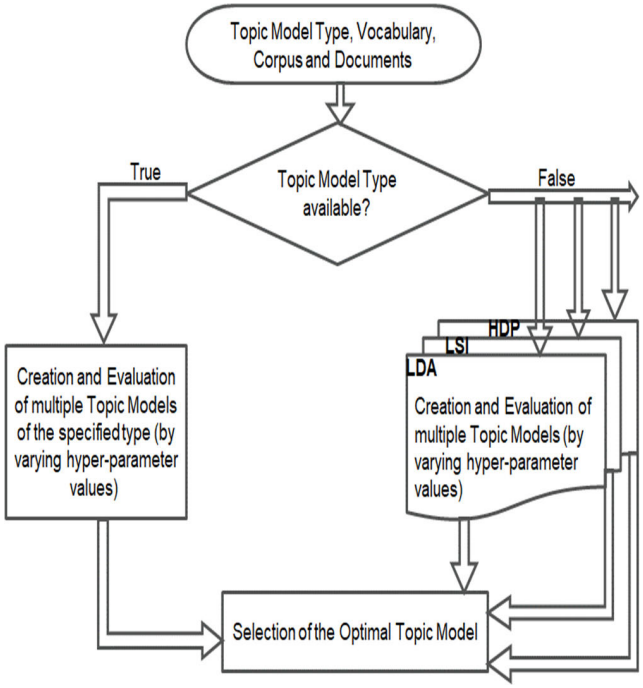


FIGURE 1. Topic model optimization and selection.

however the retraining with the proposed architecture is a unique approach for punctuation restoration. The steps of this algorithm are as follows.

1) Preprocess either *transcript* to remove *duplicate words*, *phrases* and *expressions* or *punctuations* inserted as *delimiters* based on the annotator's perceptions of the pauses in the conversation or preprocess a *summary* to remove *periods*. The output off this step is a continuous string representing the cleaned and un-punctuated text of the transcript/summary.
2) Instantiate the *pre-trained BERT punctuation model* and initialize it on GPUs to classify each encoded word in text to one of four classes, namely, 'Other': 0, 'Comma': 1, 'Period': 2 and 'Question': 3.
3) Tokenize the transcript/summary and encode *tokens* to numeric format (*token identifier/ID*) using the *BertTokenizer (bert-base-uncased).*
4) Create *segments*, of pre-specified sizes (32), of surrounding token IDs for each encoded word (token ID) from the text and insert '0' as a *placeholder* halfway through each segment and load tensor datasets (segments) of parameterized batch size (default: 512).
5) Use the placeholders from the above step to predict punctuation class identifiers (class IDs) for all token IDs in the segments using the *modified BERT* model *classifier*.
6) Map class IDs to words/symbols and merge words (if needed) to restore two sets of punctuated transcripts/summaries, one with just *periods* (partial punctuation restoration) and the other with all *punctuations* (full punctuation restoration). *Partial punctuation*

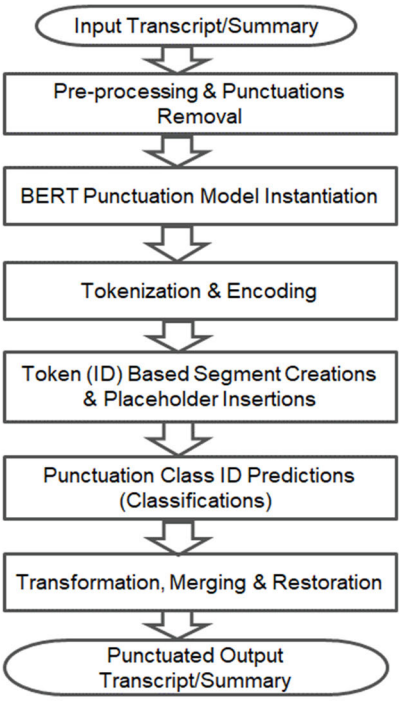


FIGURE 2. Punctuation restoration.

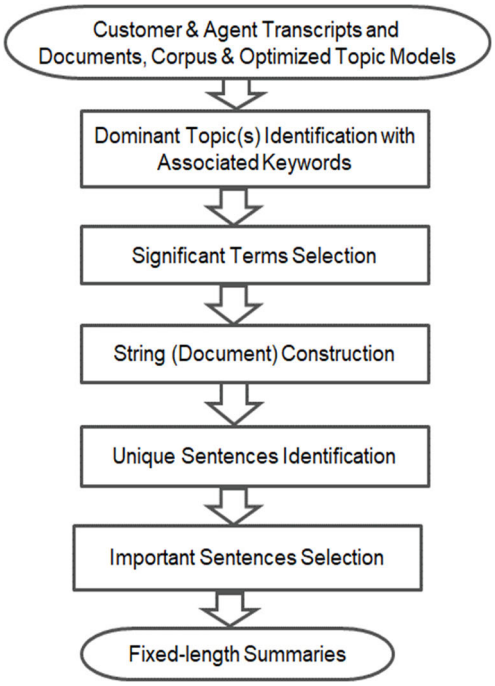


FIGURE 3. Summary generation through sentence selection.

restoration is used in step 2, while both *partial* and *full punctuation restorations* are used in step 8 of the main procedure.

Figure 2 shows the algorithm using block diagrams. We found that the BERT model for punctuation restoration provided 30% more accurate results than the LSTM based model. We implemented the punctuation restoration algorithm using *BERT Transformer*, *BertPunc* and *nn* modules, available from *PyTorch*.

E. SUMMARY GENERATION THROUGH SENTENCE SELECTION

Next, we present an algorithm for generating separate customer and agent summaries from each pair of transcripts through *sentence selection*, starting with their corresponding topic models. In other words, the following algorithm implements steps 5–7 of the proposed main procedure. The inputs to the algorithm are a pair of customer and agent transcripts and the corresponding optimized topic models, corpus, and documents from all transcripts.

1) Use selected topic models (customer and agent) to identify *dominant topic(s)* from each of the customer and agent documents for every pair and produce two lists of associated *keywords*, one for each of customer and agent transcripts. The ''number of dominant topics'' to be identified per transcript is a parameter of the procedure.
2) Use the keywords/terms associated with customer and agent dominant topics to extract the most significant inter-related *terms* for each pair of transcripts. This is achieved using *word-based similarity analysis*. As mentioned previously in the main procedure, we provide two alternatives to keyword/term extraction, where the choice is parameterized.
   If a *global extraction* is desired, then use the two lists of keywords associated with the corresponding customer and agent dominant topics in every pair, and identify *terms* that are most *similar* to each other, that is, where the degree of similarity is above a certain parameterized threshold (default: 0.5).
   Otherwise, if a *local extraction* is desired, then first find a set of *terms* from each of the customer and agent documents in every pair that is most *similar* ($\geq$ pre-specified threshold of 0.9) to the *keywords* associated with the corresponding *dominant topic*(s) for that transcript; second, identify *terms* from these two sets of terms extracted locally from customer and agent documents that are most *inter-related*, that is, where degree of similarity is above the parameterized threshold (default: 0.5).
3) Construct a *string* (or *document*) with the *extracted significant terms* for each pair of customer and agent transcripts.
4) Identify the most *unique* sentences in each of the customer and agent transcripts in every pair using embeddings and eliminate redundant sentences to condense the original transcripts. This is achieved by generating a *correlation matrix* with the embeddings for all sentences in the original transcript and removing those sentences whose correlations are above a certain parameterized threshold (default: 0.5) for uniqueness.
5) Select a certain specified number (parameter to the procedure) of the most *important* sentences from each of the condensed customer and agent transcripts in

every pair that are *most similar* to the string (for that pair) constructed in step 3 of the current procedure using *sentence-based embeddings*; list and concatenate them in order; subsequently present the results as the summaries for the corresponding transcripts.

Figure 3 illustrates the crux of this algorithm for any given pair of transcripts, using block diagrams. For *term-based similarity analysis*, we calculated *cosine similarity* between *GloVe* encoded word vectors (300 dimensions) using *spaCy's en_core_web_lg*; while for *sentence-based similarity analysis and summary generation*, we used *the Universal Sentence Encoder (USE)* from *tensorflow-hub*, along with the Python based *pandas* and *numpy* packages. The similarity between two sentences is computed as the *inner product* of the two tensors.

F. SUMMARIZATION EVALUATION

We determine the effectiveness of the summarizer by measuring both the *goodness* (*quality*) of the summarization and the *correctness* (*accuracy*) of the punctuation restoration, reflecting the readability of the summaries.

For the *quality* of the information content of the generated summaries, we can use the metric *rouge* or *bleu scores* to measure their *goodness*. We compared the customer and agent summaries against the corresponding transcripts (or manually generated summaries if available) by computing their individual *rouge* (different types) and *bleu scores* using the *Python* packages *rouge* (*Rouge*) and *NLTK* (*nltk.translate.bleu_score*).

For the *correctness* of the punctuation restoration, we define the following metric named as *punctuation-restoration-accuracy score* to measure the *accuracy* of the punctuation restoration algorithm.

Definition 1: *The punctuation-restoration-accuracy score represents the number of matches of punctuation symbols (or just periods) between the original extracted text (transcript or summary) and the punctuated text (transcript or summary), expressed as a percentage (%).*

To implement this metric, we can use the *accuracy_score* function from python's *sklearn.metrics* package. We evaluate the effectiveness of the punctuation restoration algorithm using the following two steps.

1) Extract *periods* from both the extracted customer and agent summaries (step 7 of the main procedure) as well as the *period-only* punctuation restored summaries (step 8 of the same procedure) and produce four lists, one each for *extracted* and *partially punctuated* summaries for each of the customer and agent.
2) Count the number of *matches* between the two extracted lists of *periods* from the *extracted* and *partially punctuated* summaries of both the customer and agent, and compute the above-defined accuracy scores for both. In each case the number reflects the % of *periods* from the *partially punctuated* (*periods-only*) summary (from step 8) found in the *extracted* summary (from step 7).

We computed the customer and agent *rouge, bleu* and *punctuation-restoration-accuracy* scores for every pair of customer and agent summaries, by iterating through all transcripts, and calculated their *averages* from their respective individual scores.

VI. PERFORMANCE AND EVALUATION

User satisfaction, effectiveness (quality of summaries and correctness of punctuation), efficiency (summarization time), flexibility, and performance comparison with other open-source, state-of-the-art extractive summarizers are some of the considerations that helped us evaluate the performance of our summarizer for call transcript summarization.

A. EXPERIMENTAL SETUP

We set up a Spark cluster, consisting of a driver node and dynamically allocated multiple executor nodes for data collection, preprocessing and summarization. The NVIDIA CUDA deep neural network (cuDNN v7.6) accelerated our training process for punctuation restoration. We retrained and tested the modified BERT transformer model on NVIDIA Tesla V100-SXM2-32GB GPU based nodes. The driver node used anywhere between one to four GPUs. We tested our extractive summarizer on four separate samples from four different use cases, consisting of 47, 50, 60 and 983 call transcripts respectively. The transcripts in these samples covered a wide range of issues including billing, refunds, upgrades, outages, and maintenance. The average lengths of the customer and agent transcripts in the four samples were $(4433, 6246)$, $(3722, 4773)$, $(3724, 4866)$, $(3048, 4131)$ words respectively. Evaluation of the results was both manual and automated.

B. MANUAL EVALUATION

The summaries generated by the proposed method (procedure) were manually verified and validated for content and readability by four different user groups for the four different use cases spread across multiple business units. The goal was to see if the summaries were deemed generally useful for the purposes that the transcripts were used in the specific use cases. The process was informal, and the evaluation was subjective. We relied on user (our customer) feedback and allowed the users to manage and control their own evaluation processes and satisfaction levels. Four different user groups (customers), three internal and one external, manually evaluated over 50 pairs of customer and agent summaries for four different use cases.

For user evaluations, we mainly looked for answers to questions that were both generic and specific to the use cases. The following are some examples of these two types. The answers to the questions are summaries of the corresponding feedbacks received from the different user groups.

Generic

- *Q1*: Did the customer and agent summaries in general give a fair description of the main problems (concerns) and resolutions, based on the original, unseparated call

transcripts? If so, then what % of customer and agent summaries accurately summarized the content?
A1: All four user groups felt that the customer and agent summaries generally described customer concerns and agent resolutions. If only one problem was discussed between the customer and the agent(s), then ∼80% of the call transcripts were adequately summarized and the summaries helped in a better comprehension of the information content.

- *Q2*: Did the summaries capture other secondary issues (topics) in addition to the main issue? If so then what %?
  A2: The user groups felt that only about ∼50% of the summaries captured secondary topics, if multiple issues (e.g., billing, upgrades, outage, etc.) were all discussed in the same conversation. The quality of the summaries in such cases also depended on the value assigned to the ''Number of Dominant Topics'' parameter of the procedure.
- *Q3*: Did the punctuations help in making the summaries more readable for understanding the content of the transcripts?
  A3: All four user groups unanimously opined that punctuations significantly improved the readability of the generated summaries.
- *Q4*: Have the punctuations been restored correctly?
  A4: All four user groups concluded that for *partially punctuated* summaries, ∼90% of the periods, and for *fully punctuated* summaries, ∼80% of all the punctuations were restored correctly.
- *Q5*: How did our summaries compare with manually generated summaries, if available? Note that a manually generated summary would consist of a fixed number (same as that for the automated summary) of ordered sentences extracted manually from the *period-only* customer and agent transcripts generated in step 2 of the proposed procedure that the user would deem as most important from those transcripts.
  A5: The user groups indicated that they compared the automated summaries, with ∼15 manual summaries, and found the matches to be satisfactory. Furthermore, one group observed that the accuracy was ∼80% for such a comparison if the number of sentences picked in the manual summary was approximately 8% of the whole transcript (i.e., 8 sentences for a transcript that included 100 sentences in total). However, the accuracy from such a comparison decreased if more or fewer sentences were selected for manual summaries.

Use Case Specific

- *Q1*: Would the summaries indicate the possibility of churning for callers, classified as *churners*?
  A1: In one use case, an external vendor used ∼1000 of our summaries to validate the results of their classification model that they had built for Verizon to classify potential churners. This vendor indicated that the quality of our summaries was very good, the summaries captured the negative snippets adequately and were extremely helpful in validating their results.
- *Q2*: Would the user be able to send agent summaries as short text messages to customers to prevent them from making *repeat calls*?
  A2: The relevant user group, for this specific use case, felt that very short summaries from our method (procedure), consisting of just 1-2 sentences sometimes did not capture enough of the most important information content and so might not be very successful in preventing repeat calls in these cases. Therefore, the messages either needed to be longer or the summaries (2-3 sentences long) needed to be more focused.
- *Q3*: How did the summaries generated by the proposed 10-step approach compare with summaries generated by several other open-source, off-the-shelf summarizers from transcripts that were originally ill-punctuated (with periods) and where these transcripts (for external summarizers) didn't go through an accurate period restoration step (i.e., step 2) of the proposed procedure?
  A3: Two of the user groups responded by indicating that our summaries were more meaningful and readable than the summaries generated by their existing methods, i.e., *genism summarizer*, *pytextrank*, *pysummarization autoabstractor*, *T5 (Google)*, for their use cases. The other two groups did not have existing methods as their uses cases did not directly require text summarization.

We used the feedback from each use case to improve our method and the results. We are happy to report that different business units are now using our summaries for different business purposes on an ongoing basis.

C. AUTOMATED EVALUATION

For the automated evaluation, we examined *effectiveness* and *efficiency*. We compared the performances of our summarizer with those from another very popular, open-source extractive summarizer, namely *Bert Extractive Summarizer* [58], using three pre-trained Language Models: *BERT* [36], *GPT-2* [53], and *XLNet* [54]. To measure the *effectiveness* of our summarization and to compare performances, we used the metrics *rouge-1, rouge-2, rouge-l* and *bleu scores*. We determined the *efficacy* of our punctuation restoration algorithm using our own *punctuation-restoration-accuracy score* metric.

The *efficiency* of a summarizer is important in real world applications. We measured the *efficiency* of our summarizer by recording the time taken by each of the 10 steps of the proposed method (procedure). We also compared the *efficiency* (execution time) of our *summary generation* algorithm (step 7 of the main procedure) with those of the open-source extractive summarizers by recording the time taken by each to summarize each of the four different samples.

1) RESULTS AND SUMMARIZER COMPARISONS

Table 1 and Table 2 show short (∼*5 sentences*) *summaries* from two call transcripts, after channel separations,

TABLE 1. Internet service disruption & confirmation of technician's visit.

Customer Transcript – Extracted
'she moved to a little one was thinking why not trying to understand crazy ones or players ability to change okay let me see and chop checks you all right no i mean you come up yeah i is there an a fortune was yes this is christmas cause see this is dont wanna be this yeah hes i will be all wanna actually calling because when i was thinking out sometimes problems the internet or something you know a wiring it im not sure if the internet itself is monthly somehow in knowing tina today nothing youre reading and its only if you have a clock is i see me really i never ask issue okay all over only okay all right really ok'
Customer Transcript – Summary
'I mean, you come up cause, see, do not want to be like this. Because when I was thinking out sometimes problems, the internet, or something, you know, wiring it. I am not sure if the internet itself is monthly. Somehow, in knowing there is no alternative today, nothing you can read.'
Agent Transcript – Extracted
'im gonna get your name please right see i did not would you happen to have an account number no account came in alright thank you okay she yeah this is a wait this is dsl service do you do you have a tech go out today right im sorry i got this yeah i wont let me do this youre gonna get i got to get someone from dsl for you right ill get you in their call to be just a moment'
Agent Transcript – Summary
'Am I going to get your name, please, right? Would you happen to have an account number? Do you have a tech go out today? I got to get someone from DSL for you, right? I will get you on their call, to be just a moment.'

TABLE 2. Phone and internet service disruption and confirmation of technician's visit.

Customer Transcript – Extracted
'i told the girl that i have to work in that but im on the back up you know you guys are scheduled to come out to see me tomorrow i told her the eleven and three you would not work one in five would actually work a little better i dont get off looking for i have been without phone and internet students got knows how long okay like i told her that was even feasible i know is boy you can call my cellphone its its my phone number and well cool for somebody doesnt even make sense that i should be without phone and internet service for the law shes telling me i get credit for the games i didnt use it but thats not the points i think a okay you know and neither channels good funny i said i need to see i get off work at four and im usually horrible for so that do that would make more sense than when im not moving to possibly yeah but they didnt change it yeah i guess right yes im okay i appreciate you know i need thank you no i said i just appreciate you coming back to update any yet i dont sure and thank you thank you'
Customer Transcript – Summary
'I do not get what I am looking for. I have been without phone and internet. God knows how long? Like I told her that was even feasible. Does not even make sense that I should be without phone and internet service for that long.'
Agent Transcript – Extracted
'my apologies i see the account here i see all the notes take a look at this and find out whats going on for you said unfortunately it was you said three to eleven was not good i mean levin to three what they have noted on mask on speaking with today mission each and every do need to call you back and its what number can i call you back on right im gonna go and get this verify one thing i just need to do for verification purposes i do have your account number available by chance the those last three digits thank you so much all right i see the ticket me take a look at what they have setup your let me see here still working on this year for a pre shared okay for right still working on this year for i see the order and so im trying to see we can find out whats going on its i mean i do see that levin to three still on there was what time durng the day was good for you okay let me see mean if anything you think they would remove you know i give you me to the end of the day was gonna be best for you all right so i see what im gonna do is im gonna go to our dispatch center you said release you said after for thirty crack no let me see what i can find out for you were removed at and im gonna see if we can make this like maybe the last job day or make sure that the tech doesnt right after that for thirties when youre telling us out okay pushing a brief hold i really appreciate all be right back really for really its really through it no im still working on here are you okay hold on just another moment said it was more time no not a problem the least i can do for actually wanna see like just updated time but we do i do need to verify with our dispatch im just checking one more thing should only take a moment thank you and for are you really thats true router yeah are really and good really yeah im im still working out were just waiting a little bit longer and i should have an update for you thank you okay really it really it really okay for yeah thank you really a little better thats true the a and well thank you so much for waiting all apologize for the wait theres just a cure to get in touch and some just waiting for them to come to the line i just want it off you are you okay hold a few more minutes would you rather me call you back absolutely it do you want me to call you on the number youre calling me on sure well that number for make sure havent right yep thats the one the ticket yeah what do soon as i get an update from them which i think well be in actually its gotta be very soon forty'
Agent Transcript – Summary
'I see the account here. I do have your account number available by chance. And I am going to see if we can make this, like maybe the last job day, or make sure that the tech does not go right after four thirty, when you are telling us you will be out, okay, pushing a brief hold. I really appreciate the call, and I should have an update for you. Hold a few more minutes.'

describing customer complaints about Internet and phone service disruptions and agents' confirmations about impending technicians' visits. Table 3 compares the *effectiveness* and *efficiency* of the proposed summarizer for shorter summaries with those from the open-source Bert Extractive Summarizer [35], [58], using the four different samples, on five different *evaluation metrics*. We used the Bert Extractive Summarizer with three pre-trained models, namely *bert-base-uncased* [57], *gpt2-medium* [59], *and xlnet-base-cased* [60] for summarizing the same call transcripts. Each pre-trained model was used with its own *tokenizer*, *configuration*, *vocabulary* and *checkpoints*. The gpt2-medium has 345 million parameters. Bert Extractive Summarizer generated summaries using the *period-restored* customer and agent transcripts from step 2 of the proposed procedure. Its *ratio* parameter was automatically adjusted, using the number of words in the transcript, to ensure that its summaries were of comparable (shorter) lengths. This is important as we found that the *longer* the summary, the *more similar* it was to the original transcript and the *higher* the rouge/bleu score.

In Table 3, for each sample, the *rouge (rouge-1, rouge-2, and rouge-l) and bleu scores* that are given for each type of transcript (customer/agent) represent the *average* of the rouge and bleu scores of all summaries generated from the corresponding type of transcripts contained in the sample. Likewise, for each sample, the *summarization time* represents the total time taken by the specific summarizer to generate all customer and agent summaries from the transcripts of that sample. Table 3 shows that for each of the four samples of call transcripts, 1) the proposed summarizer generated customer and agent summaries that had higher *average rouge* (all three types) and *average bleu scores, that is, overall better quality* than the summaries generated by the Bert Extractive Summarizer using the three different pre-trained models; 2) the proposed summarizer mostly took $\leq$ *half the time* compared to the Bert Extractive Summarizer, employing the three different NLP models, in summarizing all of the transcripts for that sample. Hence, the results in Table 3 establish that the

TABLE 3. Summarizer comparisons.

Sample	Size	Summarizer	Customer Rouge Score			Agent Rouge Score			Customer Bleu Score	Agent Bleu Score	Summarization Time (secs)
			Rouge-1	Rouge-2	Rouge-l	Rouge-1	Rouge-2	Rouge-l			
Sample-1	47	Proposed Summarizer	0.3	0.26	0.42	0.24	0.21	0.38	0.082	0.045	27.79
		BERT (bert-base-uncased)	0.23	0.22	0.32	0.2	0.19	0.34	0.02	0.008	55.04
		GPT-2 (gpt2-medim)	0.19	0.18	0.31	0.17	0.16	0.3	0.03	0.006	98.45
		XLNet (xlnet-base-cased)	0.21	0.21	0.32	0.19	0.18	0.34	0.02	0.009	76.42
Sample-2	50	Proposed Summarizer	0.34	0.3	0.49	0.29	0.25	0.44	0.07	0.031	22.81
		BERT (bert-base-uncased)	0.2	0.19	0.36	0.18	0.17	0.35	0.004	0.003	45
		GPT-2 (gpt2-medium)	0.17	0.16	0.31	0.14	0.14	0.30	0.01	0.001	82.67
		XLNet (xlnet-base-cased)	0.18	0.17	0.33	0.16	0.15	0.33	0.003	0.001	66.64
Sample-3	60	Proposed Summarizer	0.34	0.3	0.41	0.26	0.24	0.35	0.08	0.03	31.18
		BERT (bert-base-uncased)	0.21	0.2	0.33	0.16	0.16	0.33	0.009	0.001	56.8
		GPT-2 (gpt2-medium)	0.19	0.18	0.31	0.13	0.12	0.28	0.007	0.001	99.87
		XLNet (xlnet-base-cased)	0.2	0.19	0.33	0.15	0.14	0.31	0.012	0.001	73.54
Sample-4	983	Proposed Summarizer	0.35	0.31	0.5	0.3	0.27	0.46	0.09	0.06	341.05
		BERT (bert-base-uncased)	0.22	0.21	0.37	0.2	0.19	0.36	0.013	0.009	796.59
		GPT-2 (gpt2-medium)	0.19	0.18	0.32	0.17	0.16	0.31	0.007	0.006	1428.84
		XLNet (xlnet-base-cased)	0.2	0.19	0.34	0.18	0.17	0.33	0.009	0.007	1079.95

Evaluation Metric scores for 4 Extractive Summarizers from 4 samples of Call Transcripts

proposed summarizer was more *effective* and *efficient* than Bert Extractive Summarizer for *call transcripts*. However, this is expected for the following reasons: 1) the proposed summarizer combined *topic model optimization* with a more granular *similarity analysis-based sentence selection*, both designed specifically for call transcripts, and 2) it employed a faster, *embedding-based summary generation* step (step 7) that reduced the search space for sentence selection by removing redundant (similar) sentences from the call transcripts. Table 3 also illustrates that the BERT-based Bert Extractive Summarizer performed better than the GPT2- and XLNet-based summarizers on the *average rouge score* and the *summarization time* across all samples of call transcripts. However, on the *average bleu score*, there was no clear-cut winner amongst the three different language-model-driven BERT Extractive Summarizers. The *punctuation-restoration-accuracy scores* for customer and agent summaries varied between 90-100% in all cases.

It may also be noted that the proposed summarizer is highly parameterized and provides more options than the Bert Extractive Summarizer. However, its one limitation is that it *does not repair grammatical errors* but only reduces their numbers with fewer sentences and some post-processing.

2) LIMITATIONS OF THE RESULTS

One limitation was associated with the evaluation procedure of the proposed method. It originated from not having enough manually crafted reference summaries for the larger samples of call transcripts under consideration. In the absence of a full set of reference summaries for larger samples, we compared the generated summaries with the *period-restored* and *longer* original transcripts (from step 2 of our procedure) to compute their corresponding *rouge* and *bleu scores*. Thus, the scores were slightly lower. However, this was done for the summaries from both the proposed method (procedure) and the Bert Extractive Summarizer to ensure consistency and similarity in the comparisons, with the objective of determining the extent of overlap between the generated summaries and the original transcripts among the compared summarizers.

VII. CONCLUSION

In this paper, we presented an extractive summarization technique to address some of the challenges associated in general with call transcript summarization. We combined *channel separation*, *topic modeling* and *sentence selection* with *punctuation restoration* to generate more readable call transcript summaries to provide a better understanding of customer concerns and agent recommended solutions. This was perhaps the *first* summarizer to create and evaluate multiple types of topic models before selecting the most optimal one for summarization. We provided a fine-grained similarity analysis using both *term-based similarities* for significant term extraction and *sentence-based similarities* for extractive summarization. This similarity analysis leveraged both *GloVe*- and *USE*-based embeddings to exploit the semantic content of words and sentences to determine their significance, uniqueness and relevance. The proposed extractive summarizer was the *only* one that restored *full punctuation* to the summaries generated from either ill-punctuated or un-punctuated original call transcripts using a *novel BERT transformer-based model*. We introduced a *new metric* to evaluate the accuracy of punctuation restoration in the resulting summaries. Finally, we established the efficacy of the proposed summarizer through extensive evaluations and performance comparisons.

APPENDIX

Algorithm 1 Extractive Summarization Of Call Transcripts

Input: List of Call Transcripts, TopicModelType, NumberOfTopics, NumberOfDominantTopics, TermExtractionMethod, DesiredSummaryLength, WordSimilarityThreshold, UniquenessThreshold, SummaryTableName

Output: Lists of fully punctuated Customer and Agent Transcript summaries

Get the number of transcripts in the List of Call Transcripts (TL), i.e., $N \leftarrow$ length (TL)

Initialization of Customer & Agent Transcript and Document Lists, i.e., CTL $\leftarrow$ []; ATL $\leftarrow$ []; CDOCL $\leftarrow$ []; ADOCL $\leftarrow$ []

Initialization of Keyword & Significant Terms lists, i.e., CKL $\leftarrow$ []; AKL $\leftarrow$ []; STL $\leftarrow$ []

Initialization of Customer & Agent summary lists, i.e., CSL $\leftarrow$ []; ASL $\leftarrow$ []

Initialization of partially punctuated Customer & Agent summary lists, i.e.,
CSPL $\leftarrow$ []; ASPL $\leftarrow$ []

Initialization of fully punctuated Customer & Agent summary lists, i.e., CSFL $\leftarrow$ []; ASFL $\leftarrow$ []

For $j = 1$ to N **Do**

Read each call transcript, i.e.,
$T \leftarrow$ TL[j]

1. Separate call transcript into Customer and Agent transcripts using channel identifier (***channel separation***), i.e.,
CT, AT $\leftarrow$ splitChannel(T)

2. Preprocess transcripts (Customer & Agent) from above to remove existing punctuations and use a transformer based model to restore only "periods" as delimiters between sentences (***partial punctuation restoration* - ppr**), i.e.,
CT $\leftarrow$ ppr(CT); AT $\leftarrow$ ppr(AT)

3. Generate *documents* (list of words) from each of Customer and Agent transcripts using NLP pipeline based preprocessing (***document generation***), i.e.,
CDOC $\leftarrow$ preprocess(CT); ADOC $\leftarrow$ preprocess(CT)

Append Customer and Agent transcripts to CTL and ATL and Customer and Agent documents to CDOCL and ADOCL, i.e.,
CTL $\leftarrow$ add(CTL, CT)); ATL $\leftarrow$ add(ATL, AT));
CDOCL $\leftarrow$ add (CDOCL, CDOC);
ADOCL $\leftarrow$ add(ADOCL, ADOC)

Create separate word lists (vocabulary) and corpus (bag of words), i.e., words1, words2, corpus1 and corpus2, from each of CDOCL and ADOCL, i.e.,
words1 = dictionary (CDOCL);
words2 $\leftarrow$ dictionary (ADOCL);
corpus1 $\leftarrow$ [corpus (d) fordinCDOCL];
corpus2 $\leftarrow$ [corpus (d) fordinADOCL]

If TopicModelType **Then**

Build Customer and Agent *topic models* of "TopicModelType" using words1, corpus1, CDOCL and words2, corpus2 and ADOCL, by varying the "num_topics" hyperparameter from "NumberOfTopics" to 50 and compute *coherence scores* (c_v, etc.) for each model and output lists of topic model (tm) & score pairs (***topic model creation***), i.e.,
CTML $\leftarrow$
[tm (words1, corpus1, CDOCL, TopicModelType, n)
for n = NumberOfTopics, .., 50];
ATML $\leftarrow$
[tm (words2, corpus2, ADOCL, TopicModelType, n)
for n = NumberOfTopics, .., 50]

Else

Build Customer and Agent *topic models* of types LDA, LSI and HDP using words1, corpus1, CDOCL and words2, corpus2 and ADOCL, by varying the "num_topics" from "NumberOfTopics" to 50 and compute *coherence scores* (c_v, etc.) for each model and generate (lms) list of topic model (tm) & score pairs (***topic model creation***), i.e.,
CTML $\leftarrow \sum_{\text{TopicModelType}\varepsilon\{\text{LDA,LSI,HDP}\}}$
[tm(words1, corpus1, CDOCL, TopicModelType, n)
for n = NumberOfTopics, ..,50];
ATML $\leftarrow \sum_{\text{TopicModelType}\varepsilon\{LDA,LSI,HDP\}}$
[tm(words2, corpus2, ADOCL, TopicModelType, n)
for n = NumberOfTopics, ..,50]

4. Select the Customer and Agent topic models with the highest coherence scores, where "num_topics" is preferably close to "NumberOfTopics" (***optimal topic model selection***), i.e.,
CTM, s1 $\leftarrow$ max (CTML, key = lambda item : item[1]);
ATM, s2 $\leftarrow$ max(ATML, key = lambda item : item[1])

For $j = 1$ to N **Do**

Read each document from CDOCL and ADOCL, i.e.,
CDOC $\leftarrow$ CDOCL [j] ; ADOC $\leftarrow$ ADOCL[j]

5. Identify the top "NumberOfDominantTopics" *important topics* from the corresponding topic model and create a combined list of associated *keywords* for each of Customer and Agent documents (***dominant topic identification***), i.e.,
CDT, CDK $\leftarrow$ dominantTopic(CTM, corpus1, CDOC, NumberOfDominantTopics)
ADT, ADK $\leftarrow$ dominantTopic(ATM, corpus1, ADOC, NumberOfDominantTopics)

If *TermExtractionMethod* is "global" **Then**

Extract *significant terms* from the two lists of keywords, by comparing all pairs of keywords taking one from each list, and selecting those whose similarity ≥ "WordSimilarityThreshold" (***global significant term extraction***), i.e.,
ST ← flatten({(k1, k2) : (k1, k2) ε CDK × ADK, similarity (k1, k2) ≥WordSimilarityThreshold})

Else

Select *terms* only from (*local* to) the Customer and Agent documents that are similar to the corresponding dominant topic keywords, where the similarity ≥ 0.9, and create two lists, one for Customer and the other for Agent, i.e.,
CK ← flatten({(k1, k2) : (k1, k2) ε CDOC × CDK, similarity (k1, k2) ≥ 0.9});
AK ← flatten({ (k1, k2) : (k1, k2) ε ADOC × ADK, similarity (k1, k2) ≥0.9})

Extract *significant terms* from the above two selected lists, by comparing all pairs of keywords taking one from each list, and selecting those whose similarity ≥ "WordSimilarityThreshold" (***local significant term extraction***), i.e.,
ST ← flatten({(k1, k2) : (k1, k2) ε CK × AK, similarity (k1, k2) ≥WordSimilarityThreshold })

6. Construct a *document (string) of the significant terms* extracted in the previous step (***string/document construction***), i.e.,
ST ←′ ,′ .join(list(set(ST)))

Append Customer and Agent *keywords* to CKL and AKL and extracted *significant similar terms* to STL, i.e.,
CKL ← add (CKL, CK) ; AKL ← add (AKL, AK);
STL ← add(STL, ST)

Read each Customer and Agent transcript from CTL and ATL, i.e., CT ← CTL [*j*] ; AT ← ATL[*j*]

7a. Identify and select *unique* sentences in each of Customer and Agent transcripts, by removing all sentences from the corresponding transcript, whose similarity to the unique sentences ≥ "UniquenessThreshold" (***transcript reduction***), i.e.,
RCT← getUniqueSentences(CT,UniquenessThreshold);
RAT ← getUniqueSentences(AT,UniquenessThreshold)

7b. Select top "DesiredSummaryLength" *most important sentences* from each reduced Customer and Agent transcripts, based on similarity scores obtained from a sentence based similarity analysis between every sentence in the corresponding transcript with the *document (string) of significant terms,* to generate the corresponding Customer and Agent *summaries*, (***summary generation***), i.e.,
CS ←extractSummary(RCT, ST, DesiredSummaryLength);
AS ←extractSummary(RAT, ST, DesiredSummaryLength)

8. Restore *partial* (only "periods" - ppr) and *full punctuations* (fpr) to the generated summaries to create partially and fully punctuated Customer and Agent summaries (***punctuation restoration***), i.e.,
CSP ← ppr (CS) ; ASP ← ppr (AS);
CSF ← fpr (CS) ; ASF ← fpr(AS)

Append Customer and Agent summaries to CSL & ASL, partially punctuated Customer and Agent transcript-summaries to CSPL and ASPL and fully punctuated Customer and Agent transcript-summaries to CSFL and ASFL, i.e.,
CSL ← add (CSL, CS) ; ASL ← add (ASL, AS);
CSPL ←add (CSPL, CSP); ASPL ←add (ASPL,ASP);
CSFL ←add (CSFL, CSF) ; ASFL ←add(ASFL, ASF)

9. Save all call transcripts (TL), Customer & Agent transcripts (CTL & ATL), Customer & Agent summaries (CSL & ASL) and the corresponding partially and fully punctuated Customer and Agent transcript-summaries (CSPL, ASPL, CSFL, ASFL) in table "SummaryTableName" (STN), i.e.,
STN ← createTable(TL, CTL, ATL, CSL, ASL, CSPL, ASPL, CSFL, ASFL)

10. Compute and print average *rouge & bleu* (rb) *and punctuation-restoration-accuracy* (pra) *scores* for Customer and Agent summaries from CSPL and ASPL (***summarization evaluation***), i.e.,
crs, cbs← rb(SummaryTableName [CSPL], SummaryTableName [CTL]);
ars, abs ← rb(SummaryTableName [ASPL], SummaryTableName [ATL]);
cas ← pra(SummaryTableName [CSPL], SummaryTableName [CSL]);
aas ← pra(SummaryTableName[ASPL], SummaryTableName[ASL])

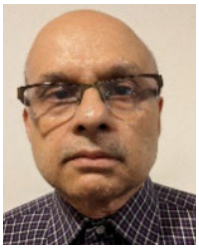
ACKNOWLEDGMENT

The authors state that they are all employees of *Verizon* and this paper addresses work performed in course of authors' employment.

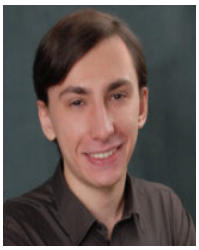
REFERENCES

[1] C. Saranyamol and L. Sindhu, "A survey on automatic text summarization," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 6, pp. 7889–7893, 2014.
[2] L. H. Reeve, H. Han, S. V. Nagori, J. C. Yang, T. A. Schwimmer, and A. D. Brooks, "Concept frequency distribution in biomedical text summarization," in *Proc. 15th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Arlington, VA, USA, 2006, pp. 604–611.
[3] D. R. Radev, E. Hovy, and K. McKeown, "Introduction to the special issue on summarization," *Comput. Linguistics*, vol. 28, no. 4, pp. 399–408, Dec. 2002.
[4] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, "Text summarization techniques: A brief survey," 2017, *arXiv:1707.02268*.
[5] M. A. Fattah and F. Ren, "Automatic text summarization," *Proc. World Acad. Sci., Eng. Technol.*, vol. 27, pp. 192–195, Feb. 2008.
[6] D. K. Gaikwad and C. N. Mahender, "A review paper on text summarization," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 5, no. 3, pp. 154–160, 2016.
[7] V. Gupta and G. S. Lehal, "A survey of text summarization extractive techniques," *J. Emerg. Technol. Web Intell.*, vol. 2, no. 3, pp. 258–268, Aug. 2010.
[8] K. Jezek and J. Steinberger, "Automatic text summarization," in *Proc. Znalosti*. Princeton, NJ, USA: Citeseer, 2008, pp. 1–12.
[9] K. S. Jones, "Automatic summarizing: The state of the art," *Inf. Process. Manage.*, vol. 43, no. 6, pp. 1449–1481, 2007.
[10] I. Mani, *Automatic Summarization*, Amsterdam, The Netherlands: John Benjamins Publishing Company, 2001.
[11] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*. Boston, MA, USA: Springer, 2012, pp. 43–76.
[12] H. Saggion and T. Poibeau, "Automatic text summarization: Past, present and future," in *Multi-Source, Multilingual Information Extraction and Summarization*. Berlin, Germany: Springer, 2013, pp. 3–21.
[13] K. Zechner, "A literature survey on information extraction and text summarization," Comput. Linguistics Program, Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., Apr. 1997. [Online]. Available: http://www.cs.cmu.edu/~zechner/infoextr.pdf
[14] H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Develop.*, vol. 2, no. 2, pp. 159–165, Apr. 1958.
[15] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, no. 2, pp. 264–285, Apr. 1969.
[16] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," in *Proc. 18th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1995, pp. 68–73.
[17] A. Bookstein, S. T. Klein, and T. Raita, "Detecting content-bearing words by serial clustering–extended abstract," in *Proc. 18th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1995, pp. 319–327.
[18] R. Brandow, K. Mitze, and L. F. Rau, "Automatic condensation of electronic publications by sentence selection," *Inf. Process. Manag.*, vol. 31, no. 5, pp. 675–685, Sep. 1995.
[19] J. M. Conroy and D. P. O'leary, "Text summarization via hidden Markov models," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2001, pp. 406–407.
[20] E. Mittendorf and P. Schauble, "Document and passage retrieval based on hidden Markov models," in *Proc. 17th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1994, pp. 318–327.
[21] F. Chen, K. Han, and G. Chen, "An approach to sentence-selection-based text summarization," in *Proc. IEEE TENCON Conf.*, Oct. 2002, pp. 489–493.
[22] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2001, pp. 19–25.
[23] D. Wang, S. Zhu, T. Li, and Y. Gong, "Multi-document summarization using sentence-based topic models," in *Proc. ACL-IJCNLP Conf.*, 2009, pp. 297–300.
[24] Z. Wu, R. Koncel-Kedziorski, M. Ostendorf, and H. Hajishirzi, "Extracting summary knowledge graphs from long documents," 2020, *arXiv:2009.09162*.
[25] N. Franciscus, X. Ren, and B. Stantic, "Dependency graph for short text extraction and summarization," *J. Inf. Telecommun.*, vol. 3, no. 4, pp. 413–429, Apr. 2019.
[26] M. Zhong, P. Liu, Y. Chen, D. Wang, X. Qiu, and X. Huang, "Extractive summarization as text matching," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 6197–6208.
[27] J. Neto, A. A. Freitas, and C. A. Kaestner, "Automatic text summarization using a machine learning approach," in *Proc. Brazilian Symp. Artif. Intell.* Berlin, Germany: Springer, 2002, pp. 205–215.
[28] K. Kaikhah, "Automatic text summarization with neural networks," in *Proc. 2nd Int. IEEE Conf. Intell. Syst.*, Jun. 2004, pp. 40–44.
[29] L. Suanmali, N. Salim, and M. S. Binwahlan, "Fuzzy logic based method for improving text summarization," 2009, *arXiv:0906.4690*.
[30] R. Nallapati, F. Zhai, and B. Zhou, "SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 3075–3081.
[31] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," in *Proc. NAACL-HLT*, New Orleans, LA, USA, 2018, pp. 1747–1759.
[32] C. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. ACL Workshop*, Barcelona, Spain, 2004, pp. 74–81.
[33] J. Xu and G. Durrett, "Neural extractive summarization with syntactic compression," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3292–3303.
[34] S. Verma and V. Nidhi, "Extractive summarization using deep learning," 2017, *arXiv:1708.04439*.
[35] D. Miller, "Leveraging BERT for extractive text summarization on lectures," 2019, *arXiv:1906.04165*.
[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[37] Y. Liu, "Fine-tune BERT for extractive summarization," 2019, *arXiv:1903.10318*.
[38] Y. Liu and M. Lapata, "Text summarization with pre-trained encoders," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3730–3740.
[39] X. Zhang, F. Wei, and M. Zhou, "HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 5059–5069.
[40] P. Lemberger, "Deep learning models for automatic summarization," 2020, *arXiv:2005.11988*.
[41] H. Lin and V. Ng, "Abstractive summarization: A survey of the state of the art," in *Proc. AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, Feb. 2009, pp. 9815–9822.
[42] A. Khan and N. Salim, "A review on abstractive summarization methods," *J. Theor. Appl. Inf. Technol.*, vol. 59, no. 1, pp. 64–72, Jan. 2014.
[43] R. Nallapati, B. Zhou, C. D. Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using Sequence-to-sequence RNNs and beyond," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, Berlin, Germany, 2016, pp. 280–290.
[44] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*.
[45] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," 2017, *arXiv:1704.04368*.
[46] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, "Generative adversarial network for abstractive text summarization," 2017, *arXiv:1711.09357*.
[47] A. Savelieva, B. Au-Yeung, and V. Ramani, "Abstractive summarization of spoken and written instructions with BERT," 2020, *arXiv:2008.09676*.
[48] G. Erkan and D. R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Intell. Res.*, vol. 22, pp. 457–479, Dec. 2004.
[49] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, Sep. 1990.
[50] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Handbook of Latent Semantic Analysis*, T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, Eds. Hillsdale, NJ, USA: Lawrence Erlbaum Associates, 2007, pp. 427–448.
[51] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[53] A. Rafdord, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[54] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," 2019, *arXiv:1906.08237*.
[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[56] D. Cer, Y. Yang, S.-Y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," 2018, *arXiv:1803.11175*.
[57] *BERT Base Model (Bert-Base-Uncased)*. Accessed: Aug. 23, 2022. [Online]. Available: https://huggingface.co/bert-base-uncased
[58] *BERT Extractive Summarizer*. Accessed: Aug. 23, 2022. [Online]. Available: https://pypi.org/project/bert-extractive-summarizer/
[59] *GPT2 Model (GPT2-Medium)*. Accessed: Aug. 23, 2022. [Online]. Available: https://huggingface.co/gpt2-medium
[60] *XLNet Base Model (XLNet-Base-Cased)*. Accessed: Aug. 23, 2022. [Online]. Available: https://huggingface.co/xlnet-base-cased



PRATIK K. BISWAS received the M.S. and Ph.D. degrees in computer science from Florida State University, USA, in 1988 and 1991, respectively. He is currently an Associate Fellow at the Artificial Intelligence and Data Center, Global Network and Technology (GNT) Division, Verizon Communications, Basking Ridge, NJ, USA. Earlier, he has held senior technical as well as managerial positions at Epic Scientific/U.S. Army CERDEC S&TCD, Scientific Research Corporation (SRC), Telcordia Research, Avaya Inc., Lucent Technologies, AT&T Bell Laboratories/AT&T, and IBM among others. He has also been a Senior Research Faculty at the Applied Research Laboratory (ARL), The Pennsylvania State University (PSU), and an Adjunct Professor of computer science at the Stevens Institute of Technology, Hoboken, NJ, USA. His current research interests include data analysis and mining, machine/deep learning, artificial intelligence, natural language processing, and recommender systems.



ALEKSANDR IAKUBOVICH received the B.S. degree in theoretical physics from Siberian Federal University, Russia, in 2015, and the M.S. degree in theoretical physics from Saint Petersburg State University, Russia, in 2017. He is currently a Principal Engineer at Core Engineering and Operations, Global Network and Technology (GNT) Division, Verizon Communications, Irving, TX, USA. Earlier, he worked as a Data Scientist at Alor Invest and AnalyticaPlus. He has seven years of experience in data and applied science as well as in developing MLOps. His current research interests include scalable AI algorithms, statistical analysis, natural language processing, computer vision, recommender systems, and deep learning.