

Extract Clinical Lab Tests From Electronic Hospital Records Through Featured Transformer Model

Lucy M. Lu, Arkansas Bioscience Institute, USA

Richard S. Segall, Arkansas State University, USA*

 <https://orcid.org/0000-0001-7627-2609>

ABSTRACT

Natural language, as a rich source of information, has been used as the foundation of the product review, the demographic trend, and the domain specific knowledge bases. To extract entities from texts, the challenge is, free text is so sparse that missing features always exist which makes the training processing incomplete. Based on attention mechanism in deep learning architecture, the authors propose a featured transformer model (FTM) which adds category information into inputs to overcome missing feature issue. When attention mechanism performs Markov-like updates in deep learning architecture, the importance of the category represents the frequency connecting to other entities and categories and is compatible with the importance of the entity in decision-making. They evaluate the performance of FTM and compare the performance with several other machine learning models. FTM overcomes the missing feature issue and performs better than other models.

KEYWORDS

Attention Mechanism, Conditional Models, Missing Features, Natural Language Processing, Transformer

1. INTRODUCTION

Natural language is the most common means of providing information to different people, to different organizations, and to different subjects; therefore, text has become a rich source of information. However, with the development of information technology, the amount of documentation has become so great that it is impossible for humans to handle. We need to take advantage of the capacity of computers to automatically understand documentation and to group documentation into different categories, topics, and subjects so that the total amount of information becomes manageable for humans.

Text mining can be applied to question answering, spam detection, semantic analysis, news categorization, and content classification, to name a few uses. Depending on the application, text data can come from different sources, such as web pages, emails, chats, social media, tickets, product

DOI: 10.4018/IJPHIMT.336529

*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

descriptions, invoices, insurance claims, user reviews, and so on. Due to the unstructured nature of the medium, it is challenging and time-consuming to extract information from text.

However, there are many open issues and challenges in text mining, such as synonyms, long-range dependencies, and multiple interacting features. Besides these challenges in the perspective of semantic analysis, we can also see issues from the perspective of data quality, such as missing features, lack of samples, and imbalanced classes. When these issues can be solved only through data modeling, not through data collection, the problem-specific data processing needs to be added to algorithms to overcome the limits on data quality. Another challenge is that the integration of domain knowledge could play an important role in text mining. Domain knowledge can help speed up text processing and increase the precision of the results. Domain-specific knowledge extraction requires semantic analysis to extract the association between the objects or concepts in the documentation. It is still challenging to make the semantic analysis efficient and scalable.

Feature engineering is an important step in machine learning. Typical text classification uses machine learning technology to perform Natural Language Processing (NLP) and to assign labels or tags to textual units, such as terms, sentences, paragraphs, documents, and queries. Normally, machine learning-based methodology performs classification in two steps: first selecting interesting features and then feeding features into classifiers to make predictions. In the training set, because the feature set plays the role of a shortcut of the context, it needs to be complete so that trained models can be used to retrieve results from new data. In other words, when the feature set is incomplete, only partial results can be retrieved by the trained models. Unlike traditional machine learning, deep learning trains word embeddings as the starting point of the classification. Feature engineering is done during model training by adjusting feature weights. However, the feature set still needs to be complete.

In terms of data modeling, text classification can be divided into two categories, one based on maximum likelihood and one based on minimum energy. For maximum likelihood-based methodology, we have Naive Bayes, Support Vector Machines (SVM), etc. For energy-based methodology, we have the Hidden Markov Model (HMM), the Conditional Random Fields model (CRF), etc. The difference between the two categories is not only in the technical details but also in how many language patterns can be modeled. Normally, maximum likelihood-based methodologies consider words to be independent tokens and use Bag of Words (BoW) to build sample sets. Minimum energy-based methodologies can fit models not only with individual words but also with the associations between words, making it possible to conduct semantic analysis. Deep learning is a separate architecture that trains word embeddings, and the classification layer is the last layer in the architecture.

Deep learning architecture is built upon neural networks. Neural approaches have the advantage of overcoming the limitations of feature engineering. Word embeddings convert input texts into an importance vector in which some words have higher significance and some words have lower significance. In this way, the words with higher significance can contribute more to the classification process and the words with lower significance can contribute less to classification process. It is optional to reduce the number of dimensions, but when the word embeddings are built, feature engineering is done.

Embedding models have a long history. The earliest embedding model is Latent Semantic Analysis (LSA; Dumais et al., 1988). LSA is based on dimension reduction. The dimensions with larger Eigenvalues are considered more significant and can be kept. The dimensions with smaller Eigenvalues are considered less significant and can be removed. The neural network model was proposed by Bengio et al. in 2001 and was based on a feed-forward neural network. However, the early embedding models underperformed classical models, so they were not generally used. In 2013 Google proposed word2vec models that were trained on billions of words and could be applied to many NLP tasks. In 2017, a contextual embedding model based on a three-layer bidirectional Long Short-Term Memory (LSTM) was trained on one billion words; this model performed better than word2vec because it could capture context and perform semantic analysis. The same year Google

developed Bidirectional Encoder Representation from Transformers (BERT). BERT consists of 340 parameters, was trained on 3.3 billion words, and is the current state-of-the-art embedding model. The trend of using pretrained larger models continues to be popular.

Although these large deep learning models show impressive performance on various NLP tasks, there are some challenges in data modeling. For example, it is hard to convert deep learning models into relational models. Linguistics as the domain knowledge in natural languages cannot be modeled in deep learning. For example, in entity resolution, we need to extract meaningful phrases from the text. The meanings of the words are defined by adjacent words and can be changed in different contexts. Some researchers argue that deep learning models do not really understand languages and are not robust enough for mission-critical domains.

Attention mechanism is an important breakthrough in deep learning. It is an increasingly popular concept and a useful tool in developing deep learning models for Natural Language Processing (NLP). Because it builds a shortcut of the context for input texts and promotes correlated words in one sentence, the importance vector in the attention layer can be updated through Markov-like updates and can be customized easily in classification. The prediction can be made by estimating how strongly the word is correlated with or attends to other words.

We propose a Featured Transformer Methodology (FTM) based on an attention mechanism. It can efficiently add domain knowledge to deep learning architecture. The attention mechanism performs Markov-like updates. When the model converts, the associations between domain features and word embeddings can be extracted.

The remainder of the paper is structured as follows: section 2 presents related work, section 3 introduces related definitions, section 4 explains the proposed methodology, section 5 evaluates the performance through experiments, and section 6 presents conclusions.

2. RELATED WORK

2.1 Feed-Forward Networks View Text as a Bag of Words

Word representation can be considered as a multiple-dimensional problem which requires high computation power. Originally, it could be solved by using a naive Bayesian logarithm of word-count ratios as feature values for SVM to train a sentiment analysis model (Wang et al., 2012). However, this methodology has been improved by deep learning by using feed-forward networks because of the high dimensional nature of neural networks.

To compute the continuous vector representation of words, two models were proposed by Mikolov et al. (2013) and were evaluated with a word similarity task: one is a BoW model; the other is a skip-n-gram model. The idea is that we removed a hidden layer for both of the two models and also defined the range for similar words for the skip-n-gram model. All these changes can efficiently decrease the computation complexity.

A global log-bilinear regression model was proposed by Mikolov et al. (2013) that combined the advantages of the two major model families in the literature: global matrix factorization and local context window methods. The model produces a vector space with meaningful substructure of 75% accuracy on a recent word analogy task, word similarity task, and entity recognition task.

Deep averaging neural network (Lyyer et al., 2015) added more averaging layers and applied dropout to improve performance.

Product quantization (Joulin et al., 2016) was used to compress word embeddings and to reduce the size of the classification models. This method could lower the memory usage to two orders of magnitude and outperformed the state of the art by a good margin in terms of compromise between memory usage and accuracy.

In Le et al. (2014), paragraph vector representations of documents were discovered to be fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. This way of constructing word vectors overcame the weaknesses of BoW by not ignoring

orders between words. It achieved a new state-of-the-art performance on several text classification and sentiment analysis tasks.

2.2 Recurrent Neural Networks (RNN)-Based Models

RNN-based models view text as a sequence of words and are intended to capture word dependencies and text structures. RNN has issues in exploding and vanishing gradients when learning long distance correlations, which can be addressed in LSTM by using memory cell to preserve state over long periods of time.

In addition to the generally used linear chain structured LSTM models, a tree structured LSTM model was proposed by Tai et al. (2015) which outperformed the linear chain structured LSTM models in semantic sentence similarity and sentiment classification.

The difference between the standard LSTM units and the Tree-LSTM units is that gating vectors and memory cell updates are dependent on the states of possibly many-child units. Additionally, instead of a single forget gate, each Tree-LSTM unit (indexed by j) contains one forget gate f_{jk} for each child k , which allows the Tree-LSTM units to selectively incorporate information from each child. For example, a Tree-LSTM model can learn to emphasize semantic heads in a semantic relatedness task, or it can learn to preserve the representation of sentiment-rich children for sentiment classification. The input word at each node depends on the tree structure used for the network. In other words, each node in the tree takes the vector corresponding to the head word as input.

The forget gate controls the extent to which the previous memory cell is forgotten, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state. The hidden state vector in an LSTM unit is therefore a gated, partial view of the state of the unit's internal memory cell. Since the value of the gating variables varies for each vector element, the model can learn to represent information over multiple time scales.

The TopicRNN (Dieng et al., 2016) model integrates the merits of RNNs and latent topic models: it captures local dependencies using an RNN and global dependencies using latent topics. Unlike previous work on contextual RNN language modeling, this model is learned end-to-end.

LSTM was extended from chain structure to tree structure in which each memory cell can reflect the history memories of multiple children cells or multiple descendant cells in a recursive process. It can provide a principled way of considering long-distance interaction over hierarchies. It outperforms the state-of-the-art recursive model by replacing its composition layers with the S-LSTM (Zhu, X., et al., 2015) memory blocks and overcoming gradient vanishing in long-distance dependency. S-LSTM can be considered as a combination of recursive neural networks and recurrent neural networks. It wires memory blocks in a partial-order structure instead of in a full-order sequence as in a chain-structured LSTM.

A machine reading simulator to render sequence-level networks can process text incrementally from left to right and perform shallow reasoning with memory and attention. The reader extends LSTM architecture by replacing the memory cell with a memory network which enables LSTM to reason about relations between tokens with a neural attention layer and then perform non-Markov state updates. Long-Short Term Memory-Network (LSTMN) (Cheng et al., 2016) outperforms KN5, RNN, and LSTM with less than 33, 21, and 7 in perplexity. In sentiment analysis, LSTMN outperforms Recursive Auto Encoder (RAE) (Socher et al., 2011), Recursive Neural Tensor Network (RNTN) (Socher et al., 2013), Dynamic Convolutional Neural Network (DCNN) (Kalchbrenner et al., 2014), Deep Recursive Neural Networks (DRNN) (Ozan et al., 2014), Convolution Neural Networks-Multichannel (CNN-MC) (Kim et al., 2014), Tubelets with Convolutional Neural Networks (T-CNN) (Kang et al., 2017), Paragraph Vector (PV) (Le et al., 2014), Constituency Tree-LSTM (CT-LSTM) (Tai et al., 2015), LSTM and two-layer LSTM with accuracy 86.3% or one-layer LSTMN and 87.0% for two-layer LSTMN. In sentence inference, LSTMN deep fusion outperforms BoW, LSTM, and match LSTM (mLSTM) (Wang et al., 2016), with 86.3% accuracy.

A general framework jointly trains a feature generator and a linear model in which the feature generator consists of region embedding + pooling (Liu et al., 2016). This framework can find an efficient way to explore a more sophisticated region embedding method using LSTM. The results show that embeddings of text regions representing complex concepts are more useful than embeddings of single words in isolation.

A region embedding method (Johnson et al., 2016; Liu et al., 2015) using LSTM can embed text regions of variable sizes, whereas the region size needs to be fixed in a CNN. Three RNN-based architectures were introduced to model text sequence with multi-task learning. The differences among them are the mechanisms of sharing information among the several tasks. Experimental results show that our models can improve the performances of a group of related tasks by exploring common features.

RNN converts representation of texts into a two-dimensional matrix (Zhu, P., et al., 2016): the time step dimension and the feature vector dimension. 2D pooling operation over the two dimensions may sample more meaningful features for sequence modeling tasks.

By using Bi-LSTM, the rich context of the whole sentence is leveraged to capture the contextualized local information in each positional sentence representation. As shown by Wan et al. (2016), by matching with multiple positional sentence representations, it is flexible to aggregate different important contextualized local information in a sentence to support the matching. Experiments on different tasks, such as question answering and sentence completion, demonstrate the superiority of our model. The model for semantic matching with multiple positional sentence representations, MV-LSTM, is defined as a sentence representation at one position. MV-LSTM can capture important local information by introducing multiple positional sentence representations. By using Bi-LSTM to generate each positional sentence representation, MV-LSTM has leveraged rich context to determine the importance of the local information. MV-LSTM performs better than ARC-I, CNTN, and LSTM-RNN. MV-LSTM obtains an 11.1% improvement over LSTM-RNN and a 5.6% relative improvement over MultiGranCNN. MV-LSTM works in three steps: (1) positional sentence representation through Bi-LSTM; (2) interactions between two sentences with cosine similarity measure, bilinear similarity measure, and tensor layer similarity measure; and (3) interaction aggregation through k-max pooling and multi-layer perception.

2.3 Attention Mechanism

In Bahdanau et al. (2016), neural machine translation has an encoder-decoder structure. It encodes a source sentence into a fixed-length vector from which a decoder can generate a translation. An automatic soft search for parts of source sentences can solve the fixed-length vector issue and help improve the translation performance.

In Graves et al. (2014), a neural Turing machine was proposed to with attentional processes through which they can be coupled to external memory resources. The combined system allows it to be efficiently trained with gradient descent.

In Vinyals et al. (2017), conditional probability of an output sequence can be learned through a neural architecture named Pointer networks. Neural attention is different from the previous attention attempts in that it uses attention as a point to select a member of input sequences as the output, instead of aligning an encoder to a decoder through an importance vector. This methodology can be used to solve the problem of variable-size output dictionaries.

2.4 Combination of Domain Knowledge

Text classification can be improved through domain-specific knowledge (Eke et al., 2021; Akhtar et al., 2020; Perevalov et al., 2021; Geetha et al., 2021) in perspectives of tokenization (Geetha et al., 2021), transfer learning (Qasim et al., 2021), and domain specific features, such as the combination of both questions and answers (Geetha et al., 2021) and the sarcasm identification (Eke et al., 2021).

3. DEFINITIONS

Attention mechanism works through a vector of importance weights in comparison with recurrent network architecture. Attention network architecture can discover connections with longer distances by using the context vector to provide shortcut connections between inputs and outputs. On the other hand, sequential models can remember a few states instead of the entire context. For example, given a word in a sentence as the input, we use the attention vector to estimate how strongly it is related to or attends to other elements in the sentence/context and then take the sum of their values weighted by the attention vector as the approximation of the output.

The context vector consumes three pieces of information: encoder hidden states, decoder hidden states, and alignment between source and target. Given a source sequence $x = [x_1, x_2, \dots, x_n]$ to generate the target sequence $y = [y_1, y_2, \dots, y_m]$, we define an attention mechanism to solve this problem.

The encoder is a recurrent network—for example, a bidirectional recurrent neural network (RNN)—with a forward hidden state \vec{h}_i^T and a backward one \overleftarrow{h}_i^T . The concatenation of the two can be used to represent the encoder state so that we can have both the preceding and the following words in the annotation of one word:

$$h_i = \left[\vec{h}_i^T, \overleftarrow{h}_i^T \right]^T, i = 1, 2, \dots, n \quad (1)$$

The decoder network has hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$ for the output word at position t , $t = 1, 2, \dots, m$, where the context vector c_t is a sum of hidden states of the input sequence, weighted by alignment scores:

$$c_t = \sum_{i=1}^n a_{t,i} h_i \quad (2)$$

$$a_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(s_{t-1}, h_i))}{\sum_{i'=1}^n \exp(\text{score}(s_{t-1}, h_{i'}))} \quad (3)$$

The alignment model assigns a score $a_{t,i}$ to the pair of input at position i and output at position t , (y_t, x_i) , on the basis of how well they match. The set of $\{a_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each input. In Bahdanau et al. (2015) the alignment score $a_{t,i}$ is parameterized by a feed-forward network with a single hidden layer, and this network is jointly trained with other parts of the model. The score function is therefore in the following form, given that \tanh is used as the non-linear activation function:

$$\text{score}(s_t, h_i) = v_a^T \tanh(W_a [s_t; h_i]) \quad (4)$$

where both v_a and W_a are weight matrices to be learned in the alignment model.

Attention mechanism can be divided into several categories: self-attention (Cheng et al., 2016), global/soft attention (Xu et al., 2015), and local/hard attention (Luong et al., 2015). Self-attention can be used to relate different positions of the same input sequence. The most popular self-attention architecture is LSTM. Global/soft attention can align attention weights over all patches, which can be done by using a window centered around the source position to compute a context vector after

aligning the position for the current target word. Local/hard attention can align attention weights one patch of the input at a time. the alignment of the attention weights is through alignment score functions, such as content-base attention, additive, location-base, general, dot-product, and scaled dot-product score functions.

The transformer model (Vaswani et al., 2017) is built on self-attention mechanism. With a sequence transduction architecture, it has an encoder-decoder structure which is aligned by attention mechanism. Attention has three applications in transformer: attention layer for encoder, attention layer for decoder, and encoder-decoder attention layer for alignment. In other words, the transformer follows neural sequence transduction architecture with both encoder and decoder, which have stacked self-attention and pointwise, fully connected layers.

Attention functions are also called alignment score functions, which can be divided into several categories according to the attention mechanism: content-based attention (Graves et al., 2014), additive attention (Bahdanau et al., 2015), location-based attention (Luong et al., 2015), general attention (Luong et al., 2015), dot-product attention (Luong et al., 2015), and scaled dot-product attention (Vaswani et al., 2017). In Vaswani et al. (2017) the transformer uses scaled dot-product attention score function.

Attention mechanism simulates hidden Markov-like updates in which each the attention point can be updated by all data points in the previous layer. If the attention functions are selected properly, when the model is converged, long distance dependencies can be obtained.

Content-based attention was inspired by the Neural Turing Machine. As the first prototype of modern computers, the Turing machine could perform all kinds of computations with a controller and an unlimited tape as memory. Content-based attention focuses on content addressing on the basis of the similarity between their current values and the values emitted by the controller. Content-based addressing can be a broader definition than location-based addressing, because contents can contain location information as well. A controller can take the values of variables and store them anywhere, retrieve them later, and perform computation. In Graves et al. (2014) the similarity measure is cosine similarity. The controller is central to the neural Turing architecture. The controller can be a feedforward network, a recurrent network, and even an LSTM. Content-based attention aligns system input to the controller output through similarity measure. In Graves et al. (2014) the similarity measure is cosine similarity.

$$score(s_t, h_i) = cosine[s_t, h_i] \quad (5)$$

Additive attention works by finding the maximum weighted sum of the output. This mechanism simulates sequence-to-sequence prediction problems. A typical application of additive attention is machine translation. The attention layer connects the encoder layer and the decoder layer on RNN or LSTM network architecture, in which conditional distribution of the sentence pairs can be learned and a corresponding translation for a given source sentence can be generated by searching for the sentence that maximizes the conditional probability. Machine translation network architecture learns, aligns, and translates simultaneously. The attention alignment function is the weighted sum of the output of the encoder.

$$score(s_t, h_i) = \sum_{j=1}^{T_x} a_{ij} h_j \quad (6)$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (7)$$

$$e_{ij} = a(s_{i-1}, h_j) \quad (8)$$

Dot-product attention can also be used to simulate sequence-to-sequence prediction problems in which long-distance dependency can be preserved as well. A typical application of dot-product attention is transformer network architecture. Transformer network architecture is built upon convolution networks instead of recurrent networks, which can allow for more significant parallelization and efficiently reduce the total amount of computation. As shown below, in Vaswani et al. (2017) the transformer used a scaled dot-product score function, which is like a dot-product score function except for a scaling factor.

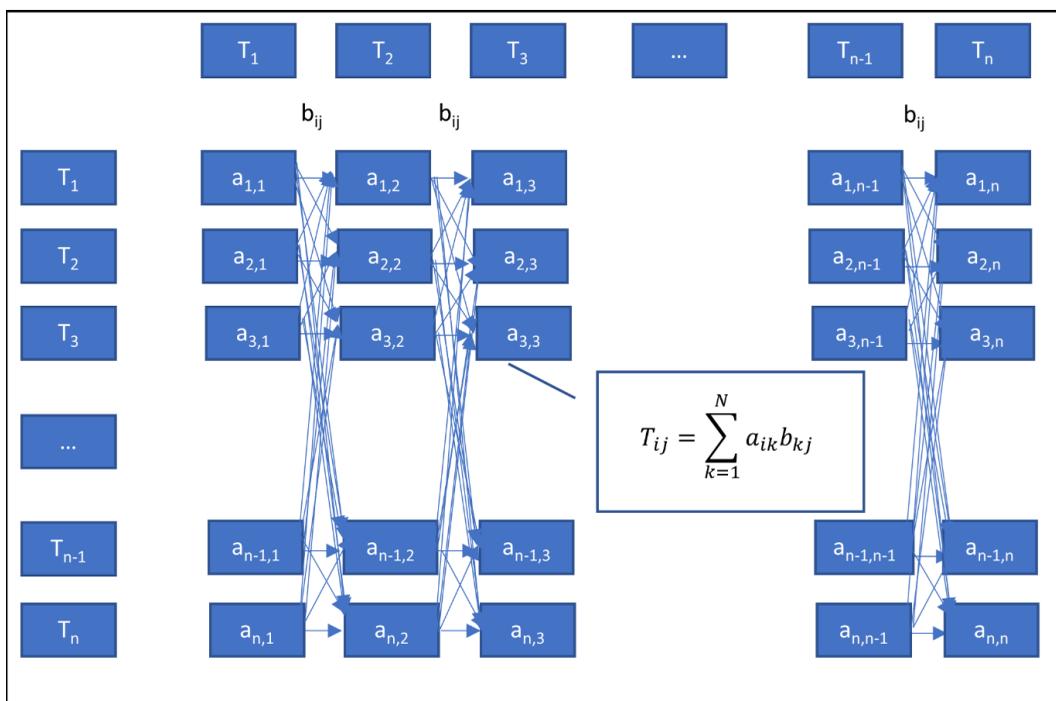
$$score(s_i, h_i) = \frac{s_i^T h_i}{\sqrt{n}} \tag{9}$$

4. METHODOLOGY

A Transformer simplifies the neural network architecture and reduces computation cost. However, several open issues have not yet been solved, such as the combination of domain-specific patterns, the combination of high-level associations, and unlabeled terms. FTM can add arbitrary features to the network architecture and can help solve feature combination, high-level association combination, and unlabeled terms issues.

BERT has transformer network architecture. Attention layer plays the role of aligning encoder and decoder layers. Encoder and decoder layers are built upon convolution networks instead of recurrent networks to reduce computation costs. The attention layer can be updated through a Markov-like process, as shown in Figure 1.

Figure 1. Markov chain updates



Connections between two sequences are presented in a two-dimensional matrix in which each dimension has a sequence of tokens. The more often the connections appear, the larger the values are. Attention mechanism is a shortcut of the Markov Chain, in which the attention layer has fewer heads than the previous layer. As shown in Figure 2, in BERT network architecture, the importance vector of the attention layer can align the encoder to the decoder and significantly affect the final prediction. If we can properly adjust the importance vector, for example, to promote positive patterns and suppress negative/random patterns, we can efficiently improve the model performance.

4.1. Model Structure

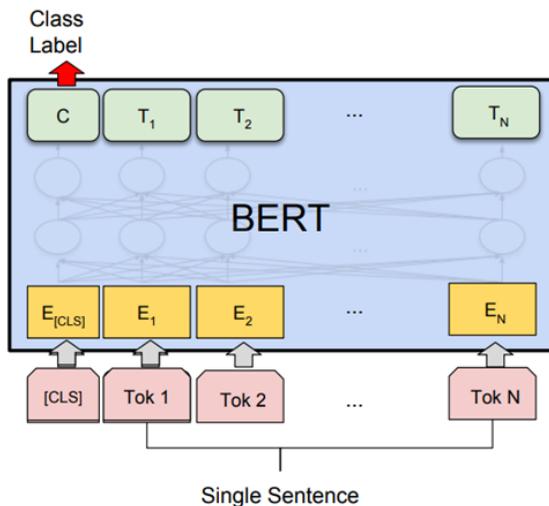
We propose Featured Transformer Methodology (FTM) to improve the performance of transformer networks, especially on BERT network architecture. The idea is that we define patterns according to the domain knowledge, add patterns as arbitrary features to the input layer, and use arbitrary features to promote the importance of corresponding terms on attention layer. FTM works in four parts, as shown below:

- (1) Input layer. We add domain-specific patterns to the input so that these patterns can be used to present the high-level features which are defined according to domain knowledge. After these patterns have been added to the input layer, they become part of the input. During model training, these patterns can be used to improve model performance. During prediction making, these patterns can contribute to the importance vector, especially for unlabeled terms that depend on these domain-specific patterns to maximize the importance of the input sequences.

Features can be defined in many ways, such as linguistic features and semantic features. For example, they can be linguistic features that include part-of-speech (POS) tags, suffixes/prefixes of the words, and others. They can also be semantic features, such as stop words, a list of abbreviations, a list of domain-specific terms, and a list of units of measurement.

The selection of the features depends on the characteristics of the data. Additional features can efficiently fix the unlabeled data issue. However, too many additional features can cover up the characteristics of the data, which may cause an overfitting problem.

Figure 2. BERT single sentence classification network architecture (Devlin et al., 2018)



- (2) BERT Encoder layer. As shown in Figure 3, the encoder layer consists of transformer blocks, which include several layers. Each layer has two sublayers: one is a multihead self-attention layer; the other is a point-wise and fully connected feed-forward network layer. The output of each sublayer is followed by layer normalization.
- (3) BERT Decoder layer. As shown in Figure 3, the decoder has a similar structure to the encoder. In addition to the two sublayers, the decoder inserts an attention layer to align the encoder output to the decoder.
- (4) Output layer. As shown in Figure 4, attention function maps a query and a set of key-value pairs to output. The outputs are computed as a weighted sum of the values where the weights are computed by compatibility function of the query with the corresponding keys.

4.2. Input Layer

Let x be an input sequence of k words, denoted as $x_{1:k} = x_1, x_2, \dots, x_k$. In BERT, an input sequence can be either one sequence or a pair of sequences separated by a special token (SEP). We use one sequence input as an example to demonstrate how the arbitrary features are added.

In Yu et al. (2019), auxiliary sentences are paired with the input to form sentence pairs so that the original classification can be converted into sentence-pair classification. Since the auxiliary sentences include higher-level common features, the classification performance can be improved after the conversion.

Figure 3. Transformer model architecture (Vaswani et al., 2017)

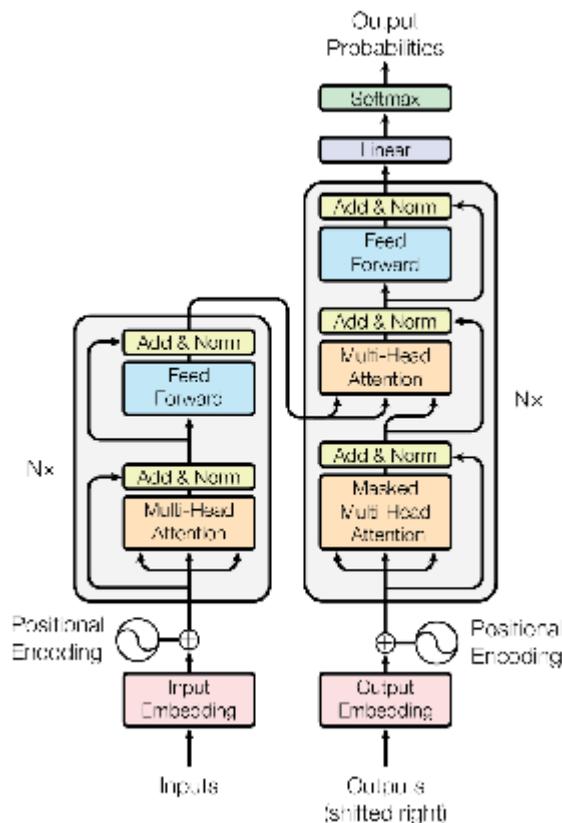
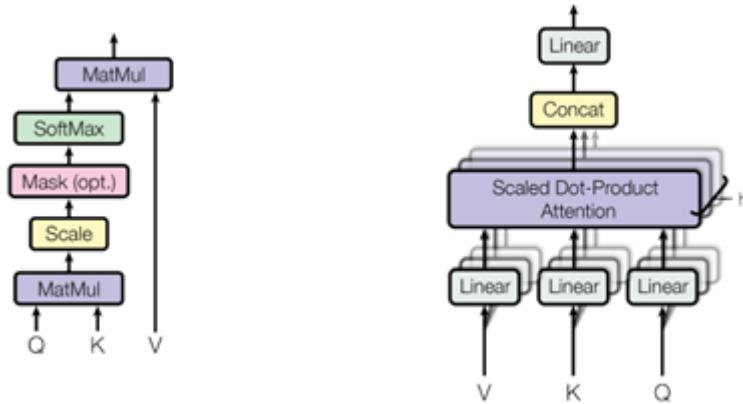


Figure 4. Attention mechanism (Vaswani et al., 2017); scaled dot-product attention (left), multi-head attention (right)



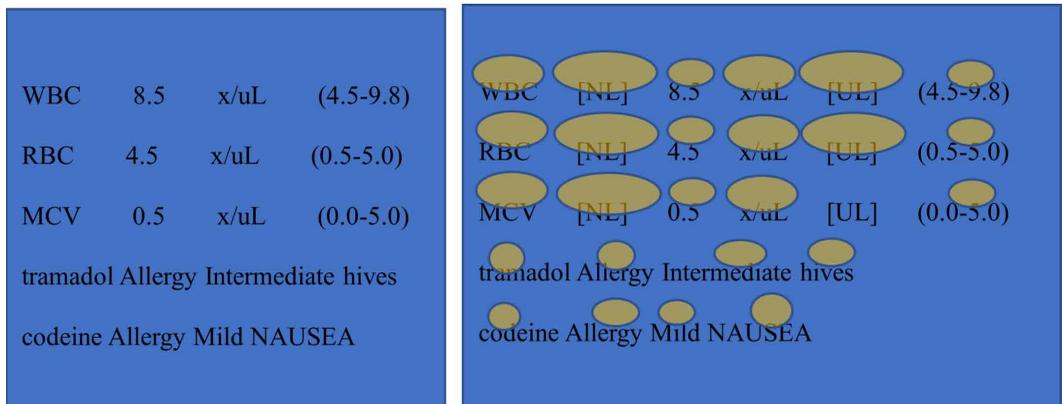
However, our idea is not to increase the accuracy of pattern matching but to increase the importance of the feature dimensions so that we directly add arbitrary features to input sequences to maximize the importance of the positive samples.

We add unit entity feature ([UL]) and name entity feature ([NL]) after unit entities and name entities. After training, as shown in Figure 5, since unit entities and name entities appear very often, unit entities and name entities can draw a high quantity of attention, which can eventually promote the importance of corresponding samples. Especially, when some entities are missing during modeling training, in prediction process, since entity features of words have been trained and obtained a high quantity of attention, the model can still count on name entity features ([NL]) and unit entity features ([UL]) to promote the importance of the input sequence.

4.3. BERT Encoder

After entity features are added to input sequences, for positive samples which have entities, entity features can gain great importance, since they are common features and appear more often. For example, as shown in Figure 5(b), feature [NL] and feature [UL] are highly related; token *WBC/RBC/MCV* and feature NL are also highly related; token *x/uL* and feature [UL] are highly related.

Figure 5. Importance of attention mechanism for input sequences: (a) original text, (b) after adding features



4.4. BERT Decoder

Entity features can be added to decoder input sequences as well. These features can pick up the importance trained at the encoder when the attention layer aligns the encoder to query sequences. When query sequences have entities unlabeled during encoding, the importance of the query can depend on entity features to get promoted. In other words, entity features can efficiently solve unlabeled feature issues.

4.5. Output Layer

The output layer is a softmax classifier on top of the attention layer. When the importance vector is tuned very well, the model can converge. During model training, the network architecture can be tuned through back-propagation, and each layer can be tuned through auto regression.

5. EXPERIMENTS

In all healthcare environments, documentation follows the Health Level Seven (HL7) Standard for electronic data exchange. For example, in the OBX segment, all the elements are defined to facilitate meeting the laboratory reporting requirements. There are a limited number of fields in the OBX segment to support compliance. For unstructured lab projects, we only use information in the fifth column, which is called OBX.5 (which means the observation 5).

In the field OBX.5, from different data sources, the formats of text contents are coded in different ways. The formats can be related to section separations, paragraph separations, sentence separations, the formats of lists, the formats of interpretations, etc. For the purpose of building a sample set of OBX laboratory tests for machine learning, we need to integrate data from different sources by converting different formats into one; after that we can start processing the data set with such operations as sample set generation, data modeling, model evaluation, and data post-processing.

For a sample selection, since we wanted to extract all lab tests (which were about 735 tests) from raw messages, we wanted to include as many lab tests as possible in the sample set. Three measurements could be used to determine sample quality: sample frequency, term frequency, and total number of terms. Sample frequency can be defined as how many samples are related to a certain term. Term frequency can be defined as how many times a certain term appears in the sample set. The total number of terms means how many distinct terms are in the sample set. During sample selection, we needed to check sample frequency and the total number of terms from time to time to ensure that the sample set covered as many terms as possible and that each term had enough samples for training purpose.

First we built a lab test dictionary that had all the terms we could extract from the raw messages. The sample selection stage could collect samples for each of the terms for model training. Each term needed between three and five samples to properly define features related to the term. If a term had fewer than three samples, the term could not be accurately extracted by the trained model. If a term had more than five samples, the sample set was over prepared.

We conducted three rounds of sample collection to prepare high-quality sample sets for data modeling purposes. In Round 1 we collected 58,746 samples, in which 22,498 samples contained lab tests, and 289 out of 735 tests were included in Round 1 samples. In Round 2 we collected 38,426 samples, in which 7,030 samples contained lab tests, and 299 out of 735 tests were included in Round 2 samples. In Round 3, we collected 21,896 samples, in which 7,067 samples contained lab tests, and 286 out of 735 tests were included in Round 3 samples. We generated two sample sets, which were the combination of Round 1 and Round 2 data, and the combination of Round 1, Round 2, and Round 3 data. For the combination of Round 1 and Round 2 data, there were 97,172 samples, in which 27,227 samples contained lab tests; 455 out of 735 tests were included in the combination of Round 1 and Round 2 samples, and 314 out of 735 tests had more than 2 samples. For the combination of Round 1, Round 2, and Round 3, there were 119,068 samples, in which 33,806 samples contained lab tests;

546 out of 735 tests were included in the combination of Round 1, Round2 and Round 3 samples, and 374 out of 735 tests had more than 2 samples.

As shown in the following table, the experimental results showed that the performance measurements, such as precision, recall, and F1-score, indicated that the model can be used to identify lab tests with 100% in all measurements. In comparison with the Conditional Random Fields (CRF) model, Bi-directional Encoder Representation Transformer (BERT), Naïve Bayes Support Vector Machine (NBSVM), Logistic Regression (LOGREG), FASTTEXT, Standard Gated Recurrent Units (STANDARD GRU), and Bi-directional Gated Recurrent Units (BiGRU), Featured Transformer Methodology (FTM) can perform much better than CRF, BERT, NBSVM, LOGREG, FASTTEXT, STANDARD GRU, and BiGRU in all measurements.

We proved the hypothesis that the selected algorithms can be used to solve the lab test extraction problem. Especially, the proposed algorithm performs better than other models. The next step is that we need to fit the algorithms with large repositories and to build models for large repositories. Some algorithms require high-performance computing machines when the data set size is large, such as BERT, Standard GRU, BiGRU, and several other deep learning algorithms. CRF is a local algorithm that works on low dimensional data and does not require high performance computing machines. In the following, we show the performance of CRF models on large repositories.

Table 1. Performance of FTM on 2871 samples

	precision	recall	f1-score
DOC	1.00	1.00	1.00
LAB	1.00	1.00	1.00
accuracy	1.00		
macro avg	1.00	1.00	1.00
weighted avg	1.00	1.00	1.00

Table 2. Performance of CRF on 2871 samples

	precision	recall	f1-score
DOC	0.927	0.989	0.957
LAB	0.959	0.764	0.85
accuracy	0.933		
macro avg	0.943	0.876	0.904
weighted avg	0.935	0.933	0.931

Table 3. Performance of BERT on 2871 samples

	precision	recall	f1-score
DOC	0.99	0.99	0.99
LAB	0.99	0.99	0.99
accuracy	0.99		
macro avg	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99

Table 4. Performance of NBSVM on 2871 samples

	precision	recall	f1-score
DOC	0.92	0.98	0.95
LAB	0.99	0.97	0.98
accuracy	0.97		
macro avg	0.96	0.97	0.96
weighted avg	0.97	0.97	0.97

Table 5. Performance of LOGREG on 2871 samples

	precision	recall	f1-score
DOC	0.92	0.96	0.94
LAB	0.98	0.97	0.97
accuracy	0.96		
macro avg	0.95	0.96	0.96
weighted avg	0.96	0.96	0.96

Table 6. Performance of FASTTEXT on 2871 samples

	precision	recall	f1-score
DOC	0.97	0.97	0.97
LAB	0.99	0.99	0.99
accuracy	0.98		
macro avg	0.98	0.98	0.98
weighted avg	0.98	0.98	0.98

Table 7. Performance of STANDARD GRU on 2871 samples

	precision	recall	f1-score
DOC	0.97	0.94	0.95
LAB	0.98	0.99	0.98
accuracy	0.997		
macro avg	0.97	0.96	0.97
weighted avg	0.97	0.97	0.97

6. CONCLUSION

We proposed a way to improve transformer architecture by adding arbitrary features to input sequences. According to the Markov-like updates, we used arbitrary features to influence the importance vector of the attention layers. This methodology can help combine domain knowledge with the input sequences,

Table 8. Performance of BiGRU on 2871 samples

	precision	recall	f1-score
DOC	0.98	0.98	0.98
LAB	0.99	0.99	0.99
accuracy	0.99		
macro avg	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99

efficiently promote the importance of the domain-specific patterns, and resolve the unlabeled data issue. We reviewed state-of-the-art solutions in transformer architecture, presented the association between Markov graph and attention mechanism, and introduced a way to implement the methodology in BERT architecture on different layers, such as the input layer, the encoder and decoder layer, the attention layer, and the output layer. We compared the new methodology—Featured Transformer—with several recently researched models, such as CRF which also uses arbitrary features for classification, BERT, NBSVM, FASTTEXT, STANDARD GRU, and BiGRU. Experimental results showed us that, according to several performance measurements, such as precision, recall, F1-score, accuracy, and micro average and weighted average precision, recall and F1-score, Featured Transformer performs better than other methodologies in text classification. In the future, we want to add lexical and semantic features and evaluate how much arbitrary features can influence data modeling.

ACKNOWLEDGMENT

Dr. Richard Segall and Dr. Shen Lu wish to acknowledge the support provided for this research by a seed money grant award from the Arkansas Biosciences Institute (ABI), as well as the support of the facilities of the Neil Griffin College of Business at Arkansas State University in Jonesboro.

REFERENCES

- Akhtar, S., Basile, V., & Patti, V. (2020). Modeling annotator perspective and polarized opinions to improve hate speech detection. *Proceedings of the Eighth AAAI Conference on Human Computation and Crowdsourcing*. doi:10.1609/hcomp.v8i1.7473
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *ICLR 2015*.
- Cheng, J., Dong, L., & Lapata, M. (2016). *Long short-term memory-networks for machine reading*. arXiv. <https://arXiv:1601.06733> 10.18653/v1/D16-1053
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv. <https://doi.org/10.48550/1810.04805>
- Dieng, A. B., Wang, C., Gao, J., & Paisley, J. (2016). *Topicrnn: A recurrent neural network with long-range semantic dependency*. ArXiv. <https://arXiv:1611.01702>
- Dumais, S. T., Furnas, G. W., Landauer, T. K., & Deerwester, S. (1988). Using latent semantic analysis to improve information retrieval. In *Proceedings of CHI'88: Conference on Human Factors in Computing* (pp. 281–285) ACM. doi:10.1145/57167.57214
- Eke, C. I., Norman, A. A., & Shuib, L. (2021). Context-based feature technique for sarcasm identification in benchmark datasets using deep learning and BERT model. *IEEE Access: Practical Innovations, Open Solutions*, 9, 48501–48518. doi:10.1109/ACCESS.2021.3068323
- Geetha, M. P., & Karthika Renuka, D. (2021). Improving the performance of aspect based sentiment analysis using fine-tuned Bert Base Uncased model. *International Journal of Intelligent Networks*, 2, 64–69. doi:10.1016/j.ijin.2021.06.005
- Graves, A., Wayne, G., & Danihelka, I. (2014). *Neural Turing machines*. arXiv. <https://doi.org/10.48550/1410.5401>
- Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé, H. III. (2015). Deep unordered composition rivals syntactic methods for text classification. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 1681–1691. doi:10.3115/v1/P15-1162
- Johnson, R., & Zhang, T. (2016). *Supervised and semi-supervised text categorization using LSTM for region embeddings*. ArXiv. <https://arXiv:1602.02373>
- Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). *Fasttext. zip: Compressing text classification models*. ArXiv. <https://arXiv:1612.03651>
- Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). *A convolutional neural network for modelling sentences*. ArXiv. <https://arXiv:1404.2188>
- Kang, K., Ouyang, W., Li, H., & Wang, X. (2017). *Object detection from video tubelets with convolutional neural networks*. CVPR.
- Kim, Y. (2014). Convolutional neural networks for sentence classification. *Proceedings of the 2014 EMNLP*, 1746–1751.
- Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. *Proceedings of the 31st ICML*, 1188–1196.
- Liu, P., Qiu, X., Chen, X., Wu, S., & Huang, X. (2015). Multi-timescale long short-term memory neural network for modelling sentences and documents. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2326–2335. doi:10.18653/v1/D15-1280
- Liu, P., Qiu, X., & Huang, X. (2016). *Recurrent neural network for text classification with multi-task learning*. ArXiv. <https://arXiv:1605.05101>
- Luong, M. T., Pham, H., & Manning, C. D. (2015). *Effective approaches to attention-based neural machine translation*. Academic Press.

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. ArXiv. <https://arXiv:1301.3781>
- Ozan, I., & Cardie, C. (2014). *Deep recursive neural networks for compositionality in language*. NIPS.
- Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543. doi:10.3115/v1/D14-1162
- Prevalov, A., & Both, A. (2021). improving answer type classification quality through combined question answering datasets. In Knowledge Science, Engineering and Management. KSEM 2021. Lecture Notes in Computer Science, vol 12816. Springer.
- Qasim, R., Bangyal, W. H., Alqarni, M. A., & Almazroi, A. A. (2022). A fine-tuned BERT-based transfer learning approach for text classification. *Journal of Healthcare Engineering*, 3498123, 1–17. Advance online publication. doi:10.1155/2022/3498123 PMID:35013691
- Socher, R., Huang, E., Pennin, J., Manning, C. D., & Ng, A. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems*, 24.
- Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., & Potts, C. (2013). *Recursive deep models for semantic compositionality over a sentiment treebank*. Academic Press.
- Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd ACL*, 1556–1566. doi:10.3115/v1/P15-1150
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *31st Conference on Neural Information Processing Systems*.
- Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., & Cheng, X. (2016). A deep architecture for semantic matching with multiple positional sentence representations. *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. doi:10.1609/aaai.v30i1.10342
- Wang, S., & Jiang, J. (2016). Learning natural language inference with LSTM. *Proceedings of the 2016 NAACL*, 1442– 1451.
- Wang, S., & Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
- Wang, Z., Hamza, W., & Florian, R. (2017). *Bilateral multi-perspective matching for natural language sentences*. ArXiv. <https://arXiv:1702.03814> 10.24963/ijcai.2017/579
- Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of the 32nd International Conference on Machine Learning*, 37.
- Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016). *Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling*. ArXiv. <https://arXiv:1611.06639>
- Zhu, X., Sobihani, P., & Guo, H. (2015). Long short-term memory over recursive structures. *Proceedings of the International Conference on Machine Learning*, 1604–1612.

Lucy Lu holds a PhD in Computer Science from University of South Florida (USF) and holds BS and MS in computer science from Tsinghua University. She is recipient of funding from a Seed Money Grant awarded by the Arkansas Biosciences Institute (ABI). Her research interests are data mining, machine learning and statistical models, and has worked on software architecture, data mining and data warehousing in several big companies. She has published many papers in text analysis, three of which have won Best Paper awards. She also won first place in Microsoft Innovation Cup Student Software Development Competition.

Richard S. Segall is Professor of Information Systems & Business Analytics in Neil Griffin College of Business at Arkansas State University in Jonesboro. He holds BS/MS in mathematics, a MS in operations research and statistics from Rensselaer Polytechnic Institute in Troy, New York, and a PhD in operations research from University of Massachusetts at Amherst. He has served on the faculty of Texas Tech University, University of Louisville, University of New Hampshire, University of Massachusetts-Lowell, and West Virginia University. His research interests include data mining, big data, text mining, web mining, database management, and mathematical modeling. His funded research includes that by U.S. Air Force, NASA, Arkansas Biosciences Institute (ABI), and Arkansas Science & Technology Authority (ASTA). He was a member of former Arkansas Center for Plant-Powered-Production (P3) and is a member of Center for No-Boundary Thinking (CNBT), serves on the editorial boards of the International Journal of Data Mining, Modelling and Management (IJDMMM), International Journal of Data Science (IJDS), and International Journal of Fog Computing (IJFC), and is co-editor of five books: (1.) Biomedical and Business Applications Using Artificial Neural Networks and Machine Learning, (2.) Open Source Software for Statistical Analysis of Big Data, (3.) Handbook of Big Data Storage and Visualization Techniques, (4.) Research and Applications in Global Supercomputing, and (5.) Visual Analytics of Interactive Technologies: Applications to Data, Text & Web Mining.