Neural Computing and Applications (2023) 35:17167–17185
https://doi.org/10.1007/s00521-023-08555-4

ORIGINAL ARTICLE

Automatic document classification via transformers for regulations compliance management in large utility companies

Tolga Dimlioglu[1] · Jing Wang[1] · Devansh Bisla[1] · Anna Choromanska[1] · Simon Odie[2] · Leon Bukhman[2] · Afolabi Olomola[2] · James D. Wong[2]

Received: 14 September 2022 / Accepted: 28 March 2023 / Published online: 28 April 2023
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2023

Abstract

The operation of large utility companies such as Consolidated Edison Company of New York, Inc. (Con Edison) typically rely on large quantities of regulation documents from external institutions which inform the company of upcoming or ongoing policy changes or new requirements the company might need to comply with if deemed applicable. As a concrete example, if a recent regulatory publication mentions that the timeframe for the Company to respond to a reported system emergency in its service territory changes from within X time to within Y time—then the affected operating groups will be notified, and internal Company operating procedures may need to be reviewed and updated accordingly to comply with the new regulatory requirement. Each such regulation document needs to be reviewed manually by an expert to determine if the document is relevant to the company and, if so, which department it is relevant to. In order to help enterprises improve the efficiency of their operation, we propose an automatic document classification pipeline that determines whether a document is important for the company or not, and if deemed important it forwards those documents to the departments within the company for further review. Binary classification task of determining the importance of a document is done via ensembling the Naive Bayes (NB), support vector machine (SVM), random forest (RF), and artificial neural network (ANN) together for the final prediction, whereas the multi-label classification problem of identifying the relevant departments for a document is executed by the transformer-based DocBERT model. We apply our pipeline to a large corpus of tens of thousands of text data provided by Con Edison and achieve an accuracy score over 80%. Compared with existing solutions for document classification which rely on a single classifier, our paper i) ensemble multiple classifiers for better accuracy results and escaping from the problem of overfitting, ii) utilize pretrained transformer-based DocBERT model to achieve ideal performance for multi-label classification task and iii) introduce a bi-level structure to improve the performance of the whole pipeline where the binary classification module works as a rough filter before finally distributing the text to corresponding departments through the multi-label classification module.

Keywords Document classification · Machine learning · BERT · Natural language processing

1 Introduction

With the rapid development of artificial intelligence (AI) [1] in recent years, computers can automatically process many tasks in industry. In the manufacturing industry [2–4], companies use robots for automatic equipment instead of human labor that frees employees from repetitive and boring tasks. AI in medicine [5–7] also becomes a hot research topic with a huge amount of breakthrough achievements in multiple directions, e.g., medical robotics [8], medical diagnosis [9, 10], medical statistics [11], human biology [12], etc. The advance in AI technology also changes the world of finance [13] by launching a hot trend of quantitative research [13]. Many Fintech companies apply machine learning methodologies in trading strategy decision [14] and high-frequency trading [15] to earn more profit. When it comes to the utility industry [16–19], people also prefer artificial intelligent technologies to minimize human intervention and save expenses.

This paper targets at a typical application scenario in the utility industry. All utility companies face the same problem: how to solve the customer requirements accurately

Tolga Dimlioglu and Jing Wang have equally contributed to this work.

Extended author information available on the last page of the article

and quickly [20, 21]. Take Con Edison as an example, tons of regulations are received every day from external regulation bodies. To resolve these regulations properly, Con Edison hires a large number of people to read the whole regulation and then forward it to the relevant departments. If the classification process could be completed automatically or semi-automatically by AI, the company could not only enhance the work efficiency and the accuracy of classification, but also save expenses in training and hiring staff (Fig. 1).

In this paper, we present an automatic document classification pipeline (shown in 1) via deep learning to solve regulation classification task at Con Edison. The pipeline consists of two parts: i) *binary classification module* aims at separating the regulations important to Con Edison from those that are not important to Con Edison and ii) *multi-label classification module* can classify the regulation important to Con Edison to specific departments within the company. Con Edison provides a large corpus of thousands of already processed regulation text data with two types of labels: i) important versus not important to Con Edison for *binary classification* task; ii) multiple labels demonstrating the specific departments to which the regulation belongs. For the multi-label classification task, the regulation might not only belong to one specific department, but it can also concern multiple departments within Con Edison. Therefore, the second task in the pipeline should be a multi-label, not multi-class classification task.

For the binary classification task, we utilize support vector machine (SVM) [22], Naive Bayes (NB) [23], random forest [24], and artificial neural network (ANN) [25] and combine the four binary soft classifiers with soft voting. The accuracy of the binary classification module in the pipeline reaches $\approx 92\%$. For the multi-label classification task, we utilize the DocBERT model (adding a fully connected ANN network after the BERT model for classification). Moreover, we apply the binary cross-entropy loss (BCELoss) instead of the classical cross-entropy loss (CELoss) since it is a multi-label not multi-class classification task. The accuracy of the multi-label classification module reaches $\approx 80\%$ under the top-3 accuracy metrics defined in Sect. 4.

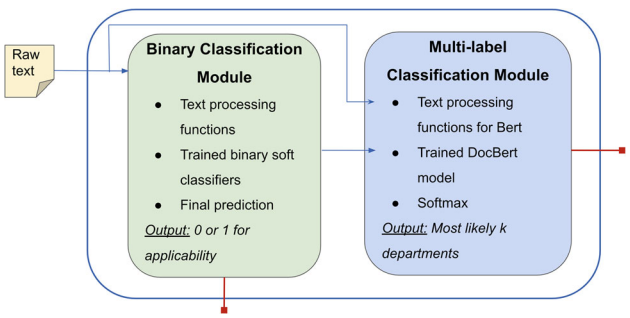

Fig. 1 Pipeline overview

This paper is organized as follows: Section 2 reviews the literature on binary classification, multi-label classification, and natural language processing. Section 3 describes the details of the datasets that we use for analysis. Section 4 defines the accuracy metrics utilized for the evaluation of the models. Section 5 discusses the construction of the automatic pipeline and its performance on the corresponding datasets. Finally, Sect. 6 concludes the contribution of our work.

2 Literature review

2.1 Binary classification

Binary classification is the process of classifying observations of a dataset into two groups based on a classification rule. It is a classical topic with multiple practical scenarios, e.g., medical testing [26], quality control in industry [27], information retrieval [28, 29], etc. Many commonly used machine learning techniques were introduced to solve binary classification problems. The Naive Bayes (NB) [23] classifier constructs the probability model based on the Bayes' theorem with strong independence assumptions between the features. Decision tree (DT) [30] is a non-parametric supervised learning algorithm that constructs a classification/regression tree by identifying ways to split a data set based on different conditions. Since a single decision tree might create over-complex trees that do not generalize well, an ensembling method, random forest (RF) [24], constructs a multitude of decision trees during the training process and then let them vote for the final results. Some other improvements in RF, e.g., AdaBoost [31], XGBoost [32], lightGBM [33], etc., became popular with desirable performance on many machine learning tasks. [22, 34] proposed the support vector machine (SVM) that constructs a hyperplane with the largest separation, or margin, between two classes. By introducing the kernel trick [22] that nonlinearly maps the inputs into a very high-dimensional space, SVM could also solve nonlinear classification problems. In 1958, psychologist Frank Rosenblatt [35] borrowed the concept of the biological neural network into computer science and proposed the first artificial neural network (ANN). The fully connected neural network is the simplest ANN where the connection between neurons in a biological neural network is modeled as weights [25]. The neural network quickly sweeps the world after being proposed due to its excellent performance in almost all application cases [36, 37]. Recently, a large amount of different types of neural networks have been proposed based on the requirements of different tasks. The

convolutional neural network (CNN) [38, 39] introduces the shared-weight architecture of the convolution kernels which slide along input features to reduce the number of parameters in fully connected neural networks, which achieve excellent performance in image processing task. In order to deal with tasks involving time-series dataset, e.g., speech recognition [40], video recognition [41], text generation [42], etc., Recurrent neural network (RNN) [43] that use previous output as inputs are purposed. To solve the drawback of forgetting long-term memory in RNN, the long short-term memory (LSTM) [44] introduces the gate construction: input gate, output gate and forget gate to the vanilla RNN to control whether to remember input of current step. Gated recurrent unit (GRU) [45] simplifies the structure of LSTM by decreasing the number of gates from 3 to 2 and achieves comparable performance in multiple tasks.

2.2 Multi-label classification

In traditional multi-class classification [46] tasks, an observation in the dataset only contains a single label from a set of labels, and we can use cross-entropy [47] as objective function. However, in multi-label classification tasks, a single observation might have multiple labels from a set of labels. The multi-label classification [48, 49] was first motivated by text classification [50] and medical diagnosis [48], where text documents contain more than one theme, and patients are prone to suffer from more than one disease. With the rapid development of technology, multi-label classification becomes essential in many modern applications, e.g., protein function classification [51, 52], music categorization [53–55], semantic scene classification [56–58], etc. The methods for the multi-label classification task could be classified into two groups: i) problem transformation methods [59–61], ii) algorithm adaptation methods [62, 63]. The problem transformation methods aim at transferring the original multi-label problem into the combination of several multi-class classification tasks. Regarding the algorithm adaptation methods, people design algorithms that could be directly applied to original multi-label task. For example, people change the cross-entropy loss for multi-class task into binary cross-entropy loss that is suitable for multi-label task [64]. However, traditional multi-label classification methods encounter many obstacles when it comes to the extreme large-scale multi-label classification problem with thousands of labels, e.g., recommendation system [65–67], natural language processing [68] and image processing [69]. Many new techniques, e.g., one versus all (OvA) classifiers [70–72], tree-based classifiers [66, 73], deep learning-based classifiers [74–76], embedding-based classifiers [77, 78], are proposed in order to solve the extreme large-scale multi-label classification task. However, our task only has hundreds of labels, which is not an extreme large-scale multi-label classification and can be solved by conventional multi-label classification methods.

2.3 Natural language processing

Languages are the most important mental creation of humans that distinguish us from animals [79, 80]. There are more than 7,100 spoken languages that exist nowadays and our connected world is filled with an abundant volume of natural language text containing different content of knowledge [81]. With the rapid advance in AI, scientists are laying more and more emphasis on the topic of natural language processing (NLP) [82–84] to enable AI to understand texts efficiently and accurately similar to humans.

The NLP technology is widely employed in many applications, e.g., speech recognition [40], sentiment analysis [85], document classification [86, 87], natural language generation [88, 89], etc. An NLP system can be separated into the following two processes: i) data processing [90] ii) model construction [84]. Data processing step [90] is aimed at mapping the text document into vectors that are understandable to the computers. Many techniques are proposed in order to vectorize long sentences or text documents by learning word associations from a large corpus of text, e.g., bag-of-words (BOW) [91], continuous bag-of-words (CBOW) [92] and skip-gram [92]. A large amount of deep learning models, e.g., convolutional neural networks (CNN) [38], recurrent neural networks (RNN) [43], textCNN [93], BiLSTM [94, 95] and attention mechanisms [96], are utilized in the model construction for NLP tasks. Recently, the emergence of a lot of powerful pre-trained models, e.g., CoVe [97], ELMo [98], OpenAI generative pre-trained transformer (GPT) [99] and bidirectional encoder representations from transformers (BERT) [100], dramatically increases the performance of deep learning models in multiple NLP tasks.

2.3.1 Transformer-based models

Transformer unit [101] was a milestone invention in NLP history and brought NLP into a new era. The self-attention mechanism proposed in transformer contains the bidirectional information of the whole text, which outperforms other sequential models like RNN, textCNN, and LSTM that only consider one-directional information in many tasks. The powerful BERT model constructed from the transformer block by taking the encoder layers is widely used in NLP tasks. RoBERTa [102] presents a replication study of BERT pretraining [100] and achieves a more powerful pretrained model by increase the training time

and batch sizes; removing the next sentence prediction objective; training on longer sequences; and dynamically changing the masking pattern applied to the training data. The decoding-enhanced BERT with disentangled attention (DeBERTa) [103] further enhances BERT by introducing the disentangled attention mechanism, incorporating absolute positions in the decoding layer to predict the masked tokens in model pre-training and using a new virtual adversarial training method to finetune. There are many BERT-based models in document classification. DocBERT [37] inserts a fully connected layer to the last hidden state vector of the BERT architecture. In the RoBERT [104] model, the hidden state vectors and posterior probabilities of the BERT model are stacked and then fed into an LSTM layer. The output of this LSTM serves as a document embedding. In ToBERT [104] model, hidden state vectors and posterior probabilities from BERT model are stacked but this time, they are fed into a transformer block since transformers are known for capturing long-distance relationships between the words in a sequence. Hierarchical attention networks (HAN) [105] are designed to capture two basic insights in the document structure. As a result, it has two levels of the attention mechanism: word level and sentence level. Words and sentences are encoded with bidirectional GRU layers, summarizing the information from both directions.

The text-to-text transfer transformer (T5) [106, 107] is a comprehensive text-to-text model designed to address various NLP tasks. Unlike other multi-task language models that rely on task-specific architectural components and loss functions, T5's creators developed a unified learning approach that treats every NLP challenge as a text-to-text problem. This enables them to use a single, consistent model, loss function, and hyperparameters to generate a unified, multi-task model. The ByT5 [108] model is a modified version of T5 that can handle text in raw byte format rather than tokens. In contrast, models such as BERT need a separate tokenization process to divide documents into sub-word vocabularies. This can result in greater memory limitations because larger vocabularies necessitate extensive embedding matrices with numerous parameters. T5-based models could also be applied in the text ranking [109] or document classification task [110, 111] in recent works. The main difference for BERT-based and T5-based models is that BERT only includes encoders, while T5 contains both encoders and decoders and perform better in natural language understanding (NLG) task. However, document classification is the natural language understanding (NLU) task not the natural language understanding (NLG) task. Therefore, focusing on BERT-based models works well for our document classification task. Moreover, T5 is a text-to-text generation model which require manually tuning the prompt [112–114]. Therefore, we focus on BERT-based model for our implementation of our work.

3 Data analysis

In this section, we provide details about the datasets provided by Con Edison. There are two dataset portions that are granted to us at different stages of the project. The first portion is provided to us for training and validation at the beginning of the project, and the held-out dataset is provided at the later stages of the project and used for testing. The dataset portions vary in terms their distributions which is mainly because of the fact that held-out dataset contains the most recent samples. Despite having different distributions, they have the same structure with two components, which are *Regulations* and *Obligations*, respectively. The connection between these two components is established with an ID key. That is, if the same ID key is associated with both a regulation text and obligation text, this implies that the text in the obligation component is the highlighted, refined fragment from the corresponding text in the regulation component.

3.1 Train and validation dataset

Here, we provide the details on the initially provided dataset which we used for training and validation. As mentioned, it has two components: *Regulations* and *Obligations*.

3.1.1 Regulations

Regulations are greater in length and some are several pages long documents including the laws or legislation put forward by the regulator state or the federal government. In this dataset, the regulations have two possible labels: *Applicable* and *Not Applicable*. As the label names imply, if a regulation label is *Applicable*, then it contains a law or legislation that is applicable to at least one of Con Edison's departments. For the regulations with *Not Applicable* label, it is vice versa. In total, there are 5570 different applicable regulations and 2212 different not applicable ones in this dataset.

3.1.2 Obligations

Since regulation texts are long, even in the applicable ones, not every part of the regulation contains the vital point of the announced law. In the *Obligations* dataset, only the important sentences or parts from the regulations are provided. Thus, *Obligations* are much shorter in length,

contain at most two paragraphs, and most of them are formed by several sentences from a single paragraph.

Note that several obligations can be deduced from the same regulation, since a regulation might contain important information in its different paragraphs or sections. Besides, a single obligation might concern more than one department of Con Edison, which makes the task carried out on the obligation dataset a multi-label classification task. Note that the labels in this dataset are the department names, anonymous here by digit numbers for confidentiality purposes.

In total, there are 111 different department names or labels, 5320 different obligation texts, and 7428 different text-label pairs. As can be seen, this is a highly imbalanced and small-sized dataset compared to the number of classes. We decided to group the departments with less than or equal to ten associated obligations into one label called *Others* since it is not really feasible for the model to learn the patterns with very few samples. By doing so, we are left with 59 different department labels including *Others* label which has 158 samples. The histogram of the dataset after this grouping method is provided in Fig. 2.

3.2 Held-out test dataset

The held-out dataset has the same structure as the train and validation dataset. This dataset contains 122 applicable and 1333 not applicable regulations. The obligations from the applicable regulations have 176 department labels. Although there are 27 different departments among the 176 department labels, six of them are grouped into the *Others* label using the dictionary obtained while forming the histogram in Fig. 2 from the training set. After this procedure, the histogram of the new set of obligations from the archived dataset is provided in Fig. 3.

4 Evaluation metrics

In total, we evaluate the models with four different metrics: accuracy (%), soft accuracy (%), Top-k accuracy (%), and normalized discounted cumulative (nDCG) score. Except for the conventional accuracy score, we introduce the remaining three metrics for evaluating the performance of experiments on the multi-label classification task.

4.1 Accuracy score (%)

Accuracy is one of the most interpretable evaluation metrics for most of the experiment results. It is simply obtained by calculating the percentage of correctly predicted labels with respect to the total size of the evaluation set. In the case of evaluating the models for the multi-label classification task, correct prediction is defined as the exact match between the target vector and the model's output layer after applying the sigmoid activation function on each of its neurons and then they are rounded to 0 or 1.

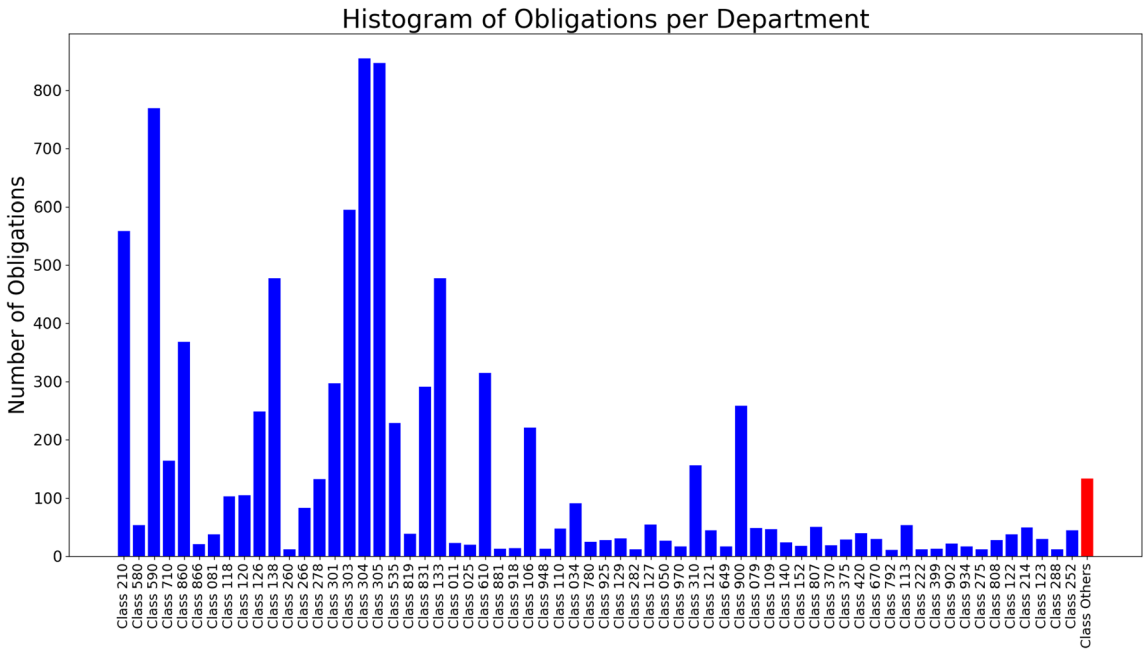

Fig. 2 Histogram of the labels in the obligation dataset (*The actual department and section names are hidden for confidentiality*)

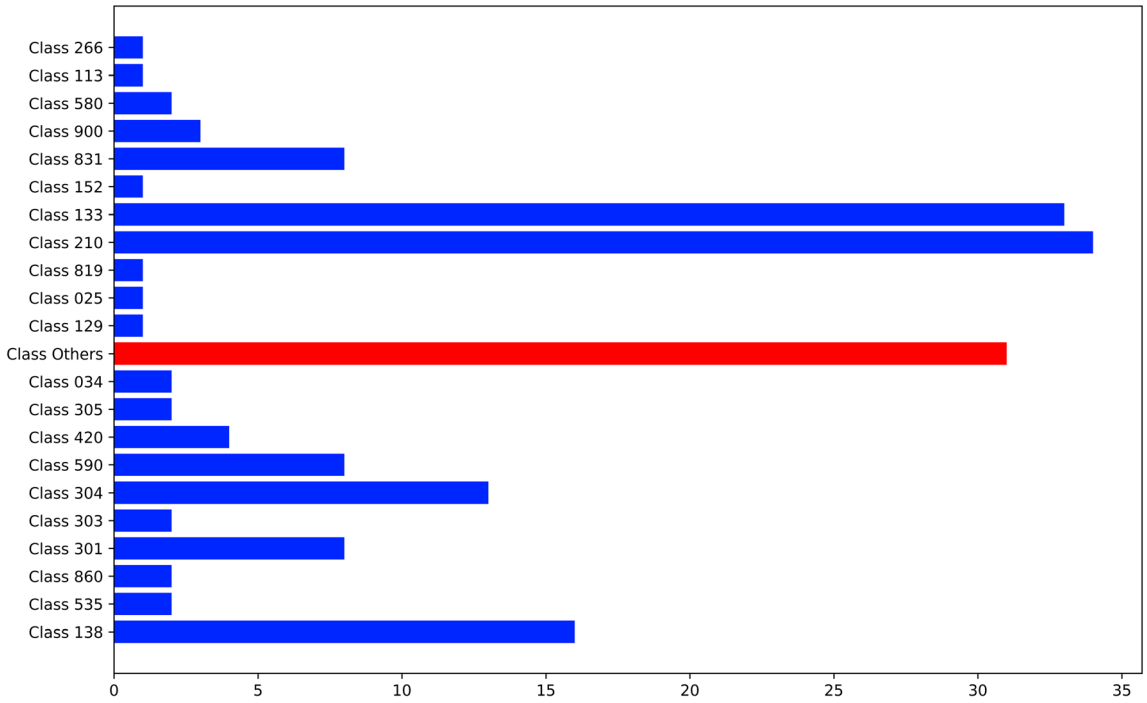

Fig. 3 Histogram of the obligations per department in the held-out set

4.2 Soft accuracy score (%)

Soft accuracy score is more generous while evaluating the performance in the multi-label classification task. Particularly, contrary to the conventional accuracy score, in this metric, the correct prediction is achieved when at least one of the sigmoid activated and rounded output layer neurons with value 1, is aligned with the target labels. In this way, the evaluation of the model is realized in a more flexible manner. Notice that this is again a percentage.

4.3 Top-*k* accuracy score (%)

Top-*k* accuracy score is a popular evaluation metric frequently used by the ML community, especially in the presence of too many target classes. In particular, if the target class belongs to the list of top-*k* most likely classes predicted by the model, the model gets the credit and this prediction is counted as correct. In the multi-label setting, if there is an overlap between the set of target classes and the set of top-*k* most likely classes predicted by the model, the prediction is counted as correct.

4.4 nDCG-*k* score

To better understand the normalized discounted cumulative gain (nDCG) score, we also need to first explain what discounted cumulative gain (DCG) score is. This is because nDCG is the normalized version of DCG. DCG quantifies the ranking success of the model prediction. Let y and $\hat{y}$ be the true label vector and the output of the classifier, respectively. That is,

$$y = [y_1, y_2, ..., y_L] \in \{0, 1\}^L$$
$$\hat{y} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_L] \in R^L$$

where L is the number of classes. Then, the DCG score is defined as follows:

$$\text{DCG-}k = \sum_{l \in \text{rank}_k(\hat{y})} \frac{y_l}{\log(l+1)}$$

DCG-k measures the accuracy based on the first k most possible classes in prediction. The term $\log(l+1)$ on the denominator controls the weights of each class. The higher the probability of a class in the prediction is, the greater the impact of the class will have on the final DCG score. Using this definition of DCG score, we normalize DCG by its log weights and formulate the nDCG score as:

$$\text{nDCG-}k = \frac{\text{DCG-}k}{\sum_{l=1}^{\min(k, \|y\|_0)} \frac{1}{\log(l+1)}}$$

5 Pipeline

In this section, we provide details of our pipeline. Particularly, the pipeline consists of two different modules: the binary classification module and the multi-label

classification module. Figure 4 shows the outline of our pipeline including the modules and their components.

The binary classification module is responsible for determining whether a given raw regulation text is applicable to Con Edison or not. If it is not applicable, the pipeline returns the result accordingly. If it is applicable, the raw text is then sent to the multi-label classification module. We would like to emphasize that the **raw** text rather than the already processed version in the binary classification module is sent to the multi-label module, since the processing steps for these two modules are different from each other. After receiving the texts, the multi-label classification module predicts the most probable k departments that the case in the regulation belongs to. Details of these modules are explained in Subsections 5.1 and 5.2.

5.1 Binary classification module

The binary classification module works as a rough filter to separate the *Regulation* dataset into two parts: the documents that are *Applicable* to Con Edison and the documents that are *Not Applicable* to Con Edison. The structure of the binary classification module shown in Fig. 5 is constructed as follows:

(1) Text processing: data cleaning and vectorize texts via bag-of-words method.
(2) Binary soft classifiers: train four binary soft classifiers: Naive Bayes (NB), support vector machine (SVM), random forest (RF), artificial neural networks with two hidden layers (ANN2).
(3) Final prediction for binary classification: ensemble binary soft classifiers by soft voting for the final prediction.

5.1.1 Text processing

After the data cleaning process of removing punctuations, eliminating numbers, and removing stopwords, the remaining text document dataset contains in total of 30733 different words. From the histogram of words shown in Fig. 6, 19108 words appear less than 10 times and 135 words appear more than 5000 times. On the one hand, the words appearing too frequently are stopwords, for example, "company," "following" and "state," which will not help with the prediction. On the other hand, the large volume of rare words without sufficient information for training drastically increases the computation cost at the same time. Therefore, we remove rare words that appear less than 10 times and frequent words that appear more than 5000 times. We get a cleaned word dictionary with 11490 remaining words.

The next step is to transfer the text information to vectors that can be mathematically processed by computers. We map the original text dataset into the vector space with bag-of-words (BOW) [92] method. The BOW method gives indexes to words in the cleaned word dictionary and records their number of occurrences in the text. Figure 7 shows an example to illustrate the BOW method.

5.1.2 Binary soft classifiers

For the binary classification task, we introduce four classical and powerful machine learning classification methods: Naive Bayes (NB) [23], support vector machine (SVM) [22, 34], random forest (RF) [24], and fully connected artificial neural network with two hidden layers (ANN2) [25]. NB classifier constructs the probability model based on Bayes' theorem with strong independence

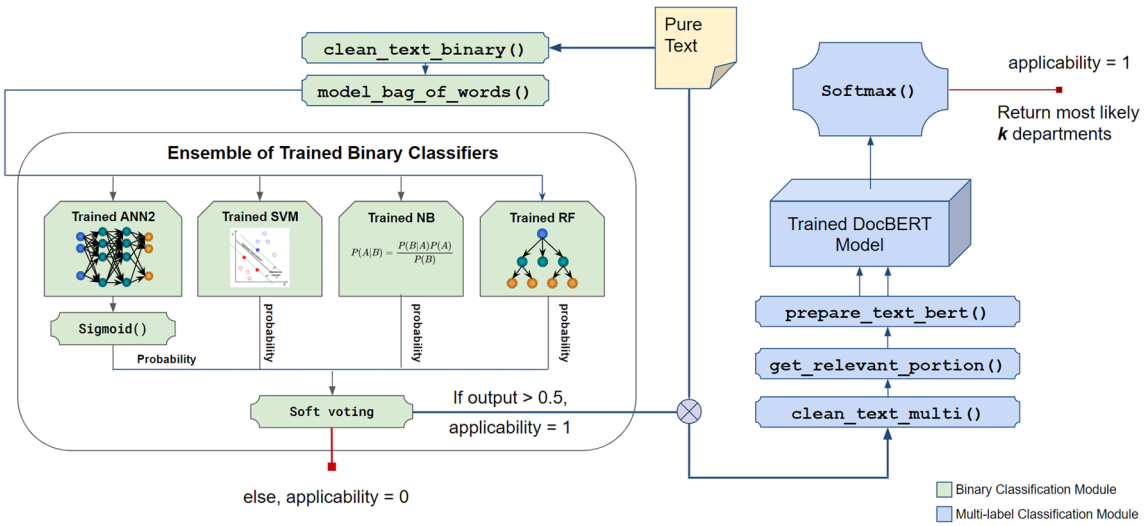

Fig. 4 Detailed diagram of the pipeline

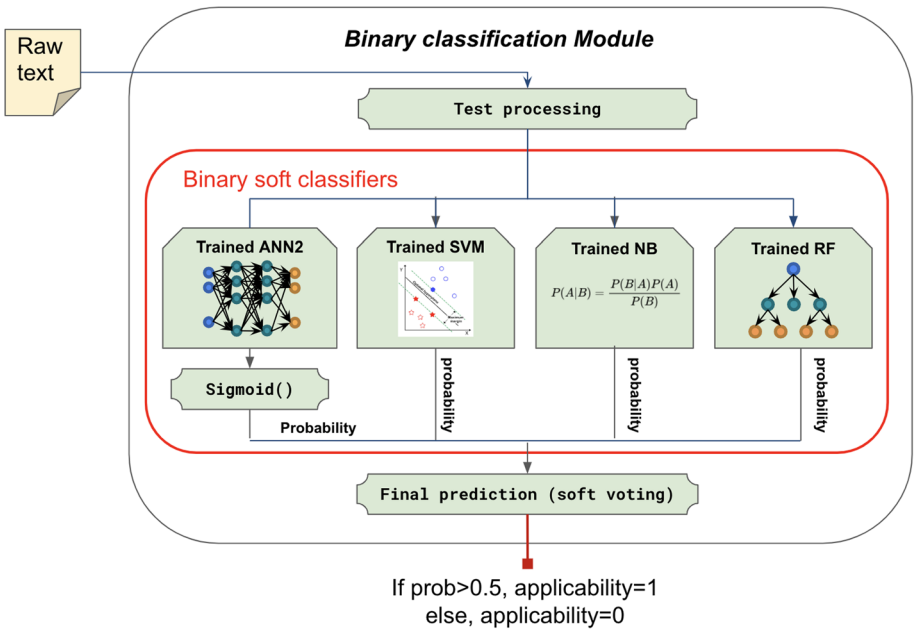

Fig. 5 Structure of binary classification module

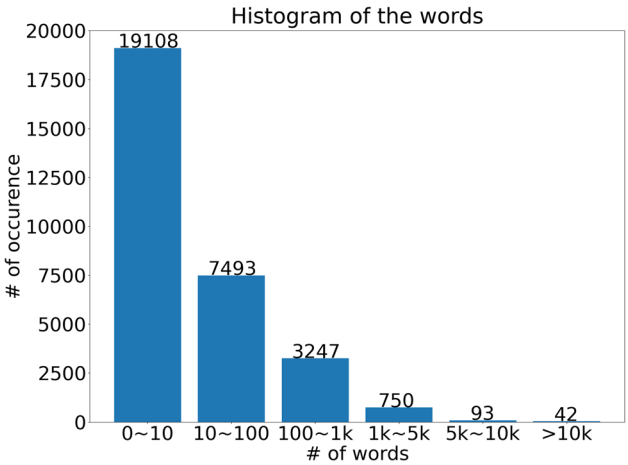

Fig. 6 Histogram of words set for binary classification task

	good	movie	not	a	did	like
good movie	1	1	0	0	0	0
not a good movie	1	1	1	1	0	0
did not like	0	0	1	0	1	1

Fig. 7 Example of the bag-of-words method

assumptions between the features. The SVM classifier separates two classes from the dataset by finding the best hyperplane with maximum margins. RF classifier constructs a multitude of decision trees during the training process, then lets them vote for the final results. For the fully connected artificial neural network classifier, we compare three network structures (Fig. 8): (i) ANN1(100) is the fully connected neural network with one hidden layer of 100 neurons; (i) ANN1(1000) is the fully connected neural network with one hidden layer of 1000 neurons; (ii) ANN2 is the fully connected neural network with two hidden layers of 1000 and 100 neurons, respectively. We perform grid search on the hyperparameters (see Table 1): initial learning rate ($lr_0 = [0.01, 0.005, 0.001]$), batch size ($bs = [16, 32, 128]$), dropout ($dp = [0.1, 0.3, 0.6]$) and weight decay ($wd = [1e{-}4, 1e{-}5, 0]$). Table 2 lists the best two settings of the parameters for each neural network. We discover that the fully connected neural network with two hidden layers where the first hidden layer has 1000 neurons and the second hidden layer has 100 neurons outperforms other net structures by $0 \sim 1\%$.

Therefore, we choose the ANN2 net structure as a soft classifier under the fully connected neural network setting and compare it with other soft classifiers: NB, SVM and RF. Table 3 illustrates the training and validation accuracy of four binary soft classifiers. We do not include the comparison of the best test accuracy among all classifiers, since there is no concept of epochs during the training procedure of NB and RF. From Table 3, each soft classifier reaches a high accuracy level of more than 85%, and ANN2 outperforms others with an accuracy of more than 97% (Table 3).

5.1.3 Final prediction for binary classification

Based on the performance of four softer classifiers illustrated in Table 20, we design two strategies for achieving the final prediction:

(1) Vanilla strategy: Utilize the single ANN2 classifier which outperforms other classifiers.

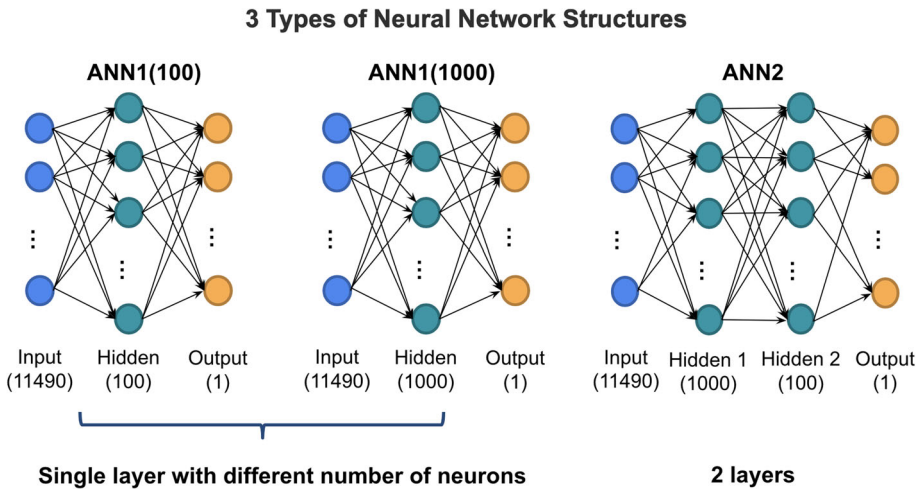

Fig. 8 Structures of three types of fully connected artificial neural networks: ANN1(100), ANN1(1000) and ANN2 from left to right

Table 1 Hyperparameter search grid for ANN models

Hyperparameter	Search grid
Batch size	16, 32, 128
Learning rate	1e−2, 5e−3, 1e−3
Dropout	0.1, 0.3, 0.6
Weight decay	0, 1e−5, 1e−4

Table 3 Training accuracy and final validation accuracy of ANN2, NB, SVM and RF

Model	Train Accu. (%)	Final Val. Accu. (%)
ANN2	99.30	**97.48**
NB	85.56	85.71
SVM	95.78	89.93
RF	**99.40**	95.65

Bold values refer to certain method is better than others

(2) Ensembling strategy: Ensemble all binary soft classifiers by soft voting. Soft voting is inspired by the voting process of random forest method. We first let each soft classifier predicts separately and then choose the class selected by most classifiers.

Intuitively, strategy #2 might be a better choice because of the following two reasons. Firstly, the test accuracy results of binary soft classifiers (Table 3) are all above 85%, which implies that even the worst NB classifier is a powerful methodology for the binary classification task. Therefore, including all the models in the final pipeline are more likely to achieve better performance. Secondly, ensembling multiple models into the final classifier could reduce variance and minimize the bias of models, and ultimately decrease the chance of overfitting. An ideal model must be able to generalize well among different datasets. However, a single soft classifier only trains on a given dataset with a fixed model structure. When it comes to a brand new dataset with not exactly the same distribution as the original dataset, the single soft classifier is quite likely to suffer from the problem of overfitting. On the contrary, ensembling multiple binary soft classifiers alleviates the risk of focusing too much on a specific feature and achieves better

Table 2 Best two settings of the parameters with maximal test accuracy after training for 200 epochs on ANN1(100), ANN1(1000) and ANN2. We compute the validation accuracy after each epoch. The final val. accu. is the test accuracy at the end of the training process. The best val. accu. is the highest test accuracy during the training process

Model	Learning rate	Batch size	Dropout	Weight decay	Train accu. (%)	Final val. accu. (%)	Best val. accu. (%)
ANN1 (100)	0.005	128	0.6	1e−5	99.20	**97.48**	97.62
	0.005	32	0.6	1e−5	99.08	97.41	97.55
ANN1 (1000)	0.005	16	0.3	0	99.44	97.21	97.41
	0.001	16	0.6	1e−5	99.15	97.14	97.28
ANN2	0.001	32	0.3	1e−5	**99.30**	**97.48**	**97.76**
	0.005	32	0.1	1e−5	**99.30**	**97.48**	97.69

Bold values refer to certain method is better than others

generalization performance. Especially for the document classification task for Con Edison, the data distribution does not stay unchanged among different time periods. Therefore, the generalization property of the model is of vital importance to achieve idea performance for our task.

In order to verify our hypothesis that strategy #2 is better, we design the following experiments to compare two prediction strategies. Except for the original regulation dataset we used for training and testing the binary soft classifiers in Sect. 5.1.2, we use the held-out test dataset to measure the generalization property of strategies on different datasets with different data distributions. The held-out test dataset is another dataset provided by Con Edison which is also the regulatory documents but from another timeline. Therefore, the data distribution is a little different from the original train and validation datasets, and it is a good fit to assess the model's robustness against the distribution shift.

For strategy #1, we simply apply the ANN2 classifier trained in Sect. 5.1.2. For strategy #2, we utilize the soft voting mechanism, which takes the average of probabilities of all binary soft classifiers as the final prediction of the whole binary classification module. Table 4 demonstrates that strategy #2 indeed generalizes better than strategy #1, and we choose strategy #2 for our final pipeline design. Although strategy #1 performs slightly better than strategy #2 on the original dataset used for training and validation, it generalizes much worse on the held-out test dataset compared with the ensembling method. Therefore, we select the second ensembling strategy when forming the final pipeline.

5.2 Multi-label classification module

The main role of the multi-label classification module of the pipeline is to list the departments that are potentially concerned by the given regulation. To do that we experiment with several models and several different configurations. First, naive ANN models are used as a baseline which takes bag-of-words as features. Then, we experiment with LSTM-based models [115], which are effective to deal with sequential data such as time series or texts. Finally, we run experiments with more sophisticated BERT-based models in our experiments. BERT [116] is a state-of-the-art language representation model that outperformed its competitors, such as OpenAI GPT, in several NLP tasks with great margins. During the experiments for this module, we have used the obligation dataset. For the experiments, 20% of the dataset is reserved for validation and the remaining 80% for training.

5.2.1 Experiments with ANN models

For this model type, similar text processing techniques are applied in the binary classification module, that is, dropping numbers and punctuation, eliminating words based on their frequency, and creating the feature vector using the bag-of-words modeling. As a result of dropping the words based on their frequency on the obligation dataset, we are left with 8638 different words. Hence, the feature vector obtained with BoW and the input size of the model is 8638. In total, we have experimented with two different ANN models varying in size:

(1) ANN1: ANN model with 1 hidden layer with 500 neurons
(2) ANN2: ANN model with 2 hidden layers with 1000 and 500 neurons, respectively.

Also, in both networks, hidden layers are followed by *ReLU* activation functions. Table 5 shows the hyperparameter search grid used in the experiments.

Hence, there are 72 different training settings for each model, and networks are trained in each setting for 300 epochs. Among all these settings, Table 7 provides the best performing results of each model type with corresponding configurations. Note that best-performing configurations are selected based on the achieved performance in conventional accuracy score on the validation set, as seen in Table 6.

The results of the Top-k accuracy scores computed on the validation set using the model configurations specified above are provided in Table 7.

Similarly, the results of the nDCG-k scores computed on the validation set using the model configurations specified above are provided in Table 8.

Table 4 Accuracy of two strategies on the original and new datasets

Strategy	Accuracy %	
	Original dataset	Held-out test dataset
Vanilla strategy	**99.02**	80.89
Ensembling strategy	96.95	**92.78**

Bold values refer to certain method is better than others

Table 5 Hyperparameter search grid for ANN models

Hyperparameter	Search grid
Learning rate	1e−3, 5e−4, 1e−4
PCA	False, 0.7, 0.8, 0.9
Dropout	0.1, 0.3
Weight decay	0, 1e−5, 1e−4

Table 6 Best performing model configurations for ANN models with soft and conventional accuracy scores

Model	LR	PCA	DP	WD	Train acc. (%)	Val. soft acc. (%)	Val. acc. (%)	Val. soft acc. (%)
ANN1	1e−3	True, 0.8	0.3	1e−5	85.46	90.12	28.22	44.07
ANN2	5e−4	True, 0.9	0.3	1e−5	88.80	94.31	29.71	44.37

Table 7 Top-k accuracy scores obtained on the validation set for ANN models

Model	Top-2 acc. (%)	Top-3 acc. (%)	Top-5 acc. (%)	Top-7 acc. (%)	Top-10 acc. (%)
ANN1	74.10	80.32	86.28	89.26	92.43
ANN2	73.29	80.06	85.34	88.59	91.72

Table 8 nDCG-k scores obtained on the validation set for ANN models

Model	nDCG-2	nDCG-3	nDCG-5	nDCG-7	nDCG-10
ANN1	0.62	0.66	0.69	0.71	0.72
ANN2	0.60	0.63	0.67	0.68	0.69

As it can be seen from Tables 6, 7, 8, although the ANN2 model achieved higher soft and classical accuracy scores, the ANN1 model reaches higher scores in terms of all the top-k accuracy and nDCG-k scores for $k = [2, 3, 5, 7, 10]$.

5.2.2 Experiments with LSTM-based models

Because BoW modeling only captures the information about the presence of the words and their frequency, it cannot incorporate the information about the context. Hence, we turn our focus toward word embeddings and models that can make use of word embeddings. Particularly, we use LSTM models since they can facilitate long-range dependencies better than RNN models due to their internal gates [115]. Because of their superiority over RNNs, the LSTMs are used as a solution to many NLP tasks such as machine translation or document classification [117].

We use a state-of-the-art LSTM-based classification model proposed in [118] which comprises a 1-layered BiLSTM network followed by a max-pooling layer on concatenated hidden states and a feed-forward network. The authors report that the proposed model achieves better accuracy scores compared to other CNN-based and LSTM-based models in text classification tasks when trained with proper regularization, hence they refer to their model as LSTM_{reg}. In accordance with the paper itself, we use dropout on hidden layers, embedding vectors, and weight decay. For the word embeddings, we use pretrained Word2Vec embeddings which capture the semantic word similarities and estimate continuous vector representations of words [119]. In Word2Vec, word embeddings are in 300 dimensional space.

The classifier part of the model is again selected from the ANN1 and ANN2 models explained in 5.2.1. For ANN1, the following hidden layer sizes are used in the experiments: [100, 500, 1000]. And for ANN2, the following hidden layer size configurations are searched as part of the hyperparameter tuning: [(500, 100), (500, 500), (1000, 100), (1000, 500)]. For both ANN1 and ANN2, the input layer size is determined in the same way and it is twice the hidden state size of the BiLSTM since the feature vector is obtained from max-pooling of the concatenated hidden states in both directions. Notice that, the hidden state size of the LSTM model is varied in the hyperparameter search. Hence, the input size of the classifier takes the values [1024, 1536, 2048] when the LSTM hidden state size is [512, 768, 1024] respectively.

We use a sequence length of 512 for the LSTM_{reg} model and if a text contains less than 512 tokens we pad it with 0s and, if it is longer than the sequence length we only take the first 512 tokens to forward to the model. We experiment with different hidden state sizes [512, 768, 1024], dropout values [0.1, 0.3] and weight decay $[1e-6, 1e-5]$ values. The LSTM_{reg} model is trained for 200 epochs with the Adam optimizer with a learning rate of $1e-4$ and a batch size of 64.

In the tables below, we provide the details and results of the best configuration that yields the smallest accuracy score for both ANN1 and ANN2 classifiers on top of the LSTM_{reg} model.

As is shown in Tables 9, 10, 11, 12, 13, the LSTM_{reg} model with the ANN1 classifier outperforms the one with the ANN2 classifier in terms of classical and soft accuracy scores. It also achieves better Top-k accuracy and nDCG-k scores.

5.2.3 Experiments with BERT-based models

Having seen the unsatisfactory accuracy scores obtained on the validation set using the ANN and LSTM models, we have also experimented with the models that use a more

Table 9 Best performing model configurations for $LSTM_{reg}$ models with soft and conventional accuracy scores

Model	Hidden Size(s)	$LSTM_{reg}$ Hidden	DP	WD	Train acc. (%)	Train soft acc. (%)	Val. acc. (%)	Val soft acc. (%)
$LSTM_{reg}$ ANN1	500	512	0.1	1e-6	77.77	87.25	31.02	50.19
$LSTM_{reg}$ ANN2	1000, 500	1024	0.3	1e-5	86.68	94.29	28.67	43.05

Table 10 Top-*k* accuracy scores obtained on the validation set for $LSTM_{reg}$ models

	Top-2 acc. (%)	Top-3 acc. (%)	Top-5 acc. (%)	Top-7 acc. (%)	Top-10 acc. (%)
$LSTM_{reg}$ ANN1	69.55	75.75	83.65	87.12	91.07
$LSTM_{reg}$ ANN2	63.38	70.87	79.98	83.83	86.94

Table 11 nDCG-*k* scores obtained on the validation set for $LSTM_{reg}$ models

Model	nDCG-2	nDCG-3	nDCG-5	nDCG-7	nDCG-10
$LSTM_{reg}$ ANN1	0.64	0.67	0.70	0.71	0.73
$LSTM_{reg}$ ANN2	0.63	0.66	0.67	0.68	0.70

Table 12 Hyperparameter search grid for the experiments with DocBERT model

Hyperparameter	Search grid
ANN1 hidden layer size	100, 500, 1000
ANN2 hidden layer sizes	(500, 500), (1000, 500)
ANN learning rate	1e−5, 1e−4
BERT learning rate	5e−7, 1e−6, 5e−6
Dropout probability	0.1, 0.3, 0.5
Weight decay	0, 1e−7

advanced model as a base, that is, the BERT model. The reason why we are going in this direction is that the feature vectors obtained with BoW modeling does not capture the contextual information at all since it only depends on the appearance of the words from the dictionary. BERT model uses transformers as its core building block; hence, it can also incorporate contextual information into the word embeddings. That is, the word bank appearing in *bank account* has a different representation than the one in *river bank*. Besides, it features bidirectional self-attention layers, meaning that it can obtain a better relation map in the attention mechanism compared to the unidirectional ones. [116]

Although the BERT model is trained on unlabeled corpus data with masked language modeling and next sentence prediction tasks, it is possible to make changes in its output layers and then fine-tune it end-to-end to make use of the BERT model's contextual understanding in any task. [116] Particularly, our task is to classify the departments to which the given regulatory case belongs, and this is a document classification task. There are several available models created by making changes in the BERT model's output layer for this type of task. Particularly, for our use case, this model is called DocBERT which is proposed in [120]. As it is explained in the paper, this

Table 13 Best performing model configurations for DocBERT models with soft and conventional accuracy scores

Model	LRs	Hidden size(s)	DP	WD	Train acc. (%)	Train soft acc. (%)	Val. acc. (%)	Val. soft acc. (%)
DocBERT ANN1	5e−6, 1e−5	100	0.1	0	99.11	99.66	46.53	61.46
DocBERT ANN2	5e−7, 1e−4	500, 500	0.3	1e−7	94.05	97.93	42.10	59.63

model is simply obtained by inserting a fully connected layer to the BERT model's first hidden state from its last layer which corresponds to the [CLS] token at the output. Note that for the classification task, insertion of the fully connected layer to this specific place is suggested by the authors of the BERT paper. [116]

For our experiments, we used two DocBERT models which replace the fully connected layer explained in the previous paragraph with ANN1 and ANN2 models. Recall that details of these models are explained in Section 5.2.1. The only change from the explained models is the input size which is now 768 instead of 8638 since the BERT model's states are 768 dimensional vectors. In total, we experimented with DocBERT models in four different schemes:

- DocBERT with ANN1 classifier using Adam optimizer
- DocBERT with ANN2 classifier using Adam optimizer
- DocBERT with ANN1 classifier using Adagrad optimizer
- DocBERT with ANN2 classifier using Adagrad optimizer

Even though we run experiments with different hyperparameters in the listed schemes, we observe that the Adagrad optimizer does not converge. The motivation for using the Adagrad optimizer is that we want to finetune the parts of the BERT and classifier weights that are not updated that frequently. And similarly, we prefer not to change the well-updated parts that much. Although this is the main strength of using the Adagrad optimizer, it drastically underperformed compared to the Adam optimizer. Thus, we do not bother sharing the results obtained with the Adagrad optimizer. In Table 12, we share the hyperparameter search grid for the experiments with DocBERT models.

Then, in Table 13, we also share the highest accuracy scores achieved in both schemes (DocBERT with ANN1 and ANN2) in addition to the hyperparameter selections yielding these results. Again, the best-performing settings are selected based on the performance of models on the validation set. Finally, we also provide the results with Top-k accuracy scores and nDCG scores in Table 14.

As is shown in Tables 13, 14, 15, the DocBERT model with the ANN1 classifier outperforms the one with the ANN2 classifier in terms of classical and soft accuracy scores, whereas it obtains inferior results in Top-k accuracy and nDCG-k scores.

5.2.4 Multilabel classification final remarks

In Table 16, we share the results of best-performing ANN, LSTM_{reg} and DocBERT models based on their conventional accuracy scores and top-k accuracy scores on the validation set.

As can be seen, the transformer-based BERT variant DocBERT model is better than LSTM_{reg} and *ANN* models in terms of accuracy scores. DocBERT model attains around 17% and 15% more accuracy than the best performing ANN and LSTM_{reg} models, respectively. The accuracy metric essentially highlights the confidence of the model in exactly predicting the ground truth labels which indicates how much generalization capability the model acquired in a certain manner. The superiority of DocBERT is also present in the other metrics as well. This also demonstrates the effectiveness of the transformers in handling long-range dependencies and extracting semantic information. Hence, we use the DocBERT model with the ANN1 classifier in our multilabel classification module, as part of our pipeline.

5.3 Pipeline results

After combining the binary and multi-label modules as it is shown in Fig. 1, we obtain the pipeline for document elimination and classification. Then, we evaluate the performance of our pipeline on both the previous train + val set and the held-out test set (Table 17). Note that we only report the classical accuracy for the binary classification module and the top-k accuracy score for the multi-label classification module since these are the most interpretable ones. Regarding the top-k accuracy score, we only report for $k = [2, 3, 5]$ since these are selected as the most

Table 14 Top-k accuracy scores obtained on the validation set with the same DocBERT models

Model	Top-2 acc. (%)	Top-3 acc. (%)	Top-5 acc. (%)	Top-7 acc. (%)	Top-10 acc. (%)
DocBERT ANN1	74.34	80.00	84.72	87.64	91.10
DocBERT ANN2	75.21	81.23	88.32	90.52	93.14

Table 15 nDCG-k scores obtained on the validation set with the same DocBERT models

Model	nDCG-2	nDCG-3	nDCG-5	nDCG-7	nDCG-10
DocBERT ANN1	0.65	0.68	0.71	0.73	0.74
DocBERT ANN2	0.65	0.69	0.72	0.74	0.75

Table 16 Comparison of best performing multilabel classification models

	Acc. (%)	Soft acc. (%)	Top-2 acc. (%)	Top-3 acc. (%)	Top-5 acc. (%)
ANN	29.71	44.07	74.10	80.06	85.34
$LSTM_{reg}$	31.02	50.19	69.55	75.75	83.65
DocBERT	46.53	61.46	74.34	80.00	84.72

Table 17 Result of the pipeline and its components on train + validation set

	Accuracy		
	$k = 2$	$k = 3$	$k = 5$
Binary module	96.95		
Multi-label module	75.43	79.25	84.32
Pipeline	79.69	82.40	85.95

reasonable k values by the Con Edison team at the operational level.

Although it might be insignificant to share the results on the train + validation set for the binary classification module, we would like to emphasize the accuracy scores achieved on the multi-label classification part and the overall pipeline. This is mainly because the evaluation of the pipeline and its components are carried out with the regulations, meaning that the multi-label classification part has not exactly seen this data but only some sections of its text. The results show that there is not much deviation from the results obtained on the validation portion of the obligation dataset.

We also test this pipeline on the held-out test set. Table 18 shows the pipeline results obtained on this dataset.

Table 18 Result of the pipeline and its components on the held-out set

	Accuracy		
	$k = 2$	$k = 3$	$k = 5$
Binary module	92.78		
Multi-label module	50.00	54.91	60.66
Pipeline	76.91	77.32	77.80

Table 17 and Table 18 together illustrate that there is a drop in the accuracy scores of the multi-label module and the overall pipeline. It is mainly caused by a large number of data samples with only *Others* as its associated label in the new held-out test dataset. However, this is not the case in the training set, which implies that there is a change in the distribution of the new held-out dataset compared with the previous one. We also assess the pipeline by ignoring these samples and provide obtained results in Table 19. There is a considerable increase in the performance of the multi-label classification module since the distributions between the train set and held-out set become closer to each other. Overall, the pipeline achieves success in both determining whether a regulation is applicable to Con Edison or not and, finding the most related departments that are concerned by the applicable regulations.

6 Conclusion

Our research proposes a deep learning-based automatic document classification system that aims to efficiently allocate text regulations to the relevant departments in Con Edison. The system achieves high accuracy scores, with over 90% accuracy for binary classification and over 80% Top-3 accuracy for multi-task classification on the given datasets. The pipeline can be used in various contexts where document classification is required, but it has a limitation in classifying long documents, as the DocBERT model used for multi-label tasks can only process documents with fewer than 512 tokens. Currently, the system only considers the last 512 tokens of a long document for multi-label classification, which may not be suitable for applications with much longer documents. To overcome this issue, we plan to incorporate embedding or text abstraction techniques in our pipeline for long document processing in the future.

Table 19 Result of the pipeline and its components on the new held-out set without the samples that have only *Others* label

	Accuracy		
	$k = 2$	$k = 3$	$k = 5$
Binary module	96.95		
Multi-label module	62.89	69.07	76.29
Pipeline	94.10	94.61	95.21

Appendix A Detailed Pipeline Figure

In the figure below, the detailed diagram for the pipeline is provided. See Fig. 9.

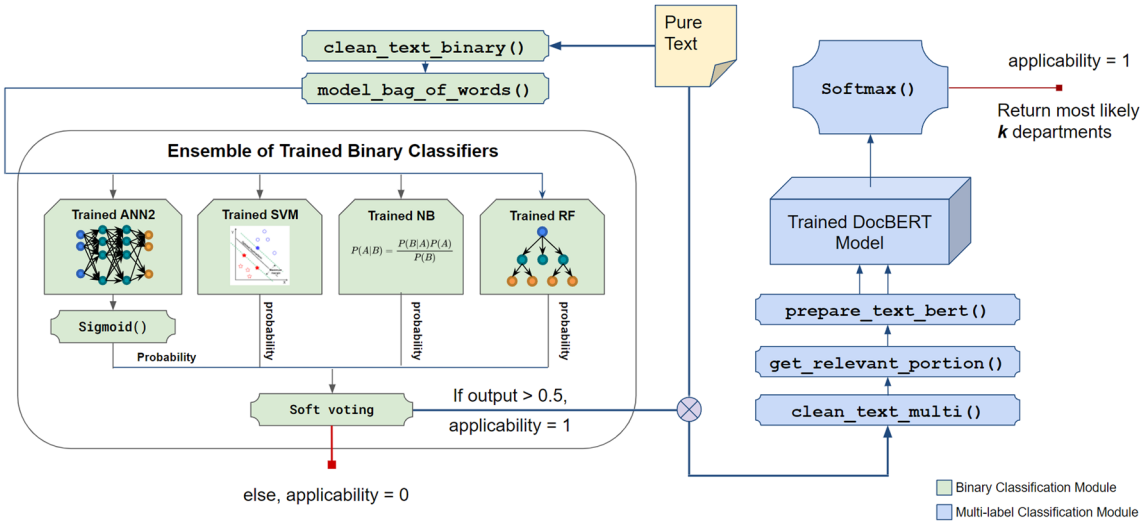

Fig. 9 Detailed diagram of the pipeline

Appendix B Binary classifier comparison on different dataset

We provide the results for four soft classifiers not only with train and validation dataset accuracy (shown in Table 20), but also the held-out test dataset. Even though SVM perform better on the held-out test dataset, it could not outperforms other classifiers on the original train and validation dataset. Moreover, not a single soft classifier could beat the final ensembling strategy on the held-out test dataset provided in Section 5.1.3 with held-test accuracy 92%.

Table 20 Training accuracy and final validation accuracy of ANN2, NB, SVM and RF

Model	Train accu. (%)	Final val. accu. (%)	Held-out test accu. (%)
ANN2	99.30	**97.48**	80.89
NB	85.56	85.71	75.35
SVM	95.78	89.93	**88.65**
RF	**99.40**	95.65	61.85

Bold values refer to certain method is better than others

Data availability The datasets analyzed during the current study are not publicly available due to commercial and privacy restriction since the datasets contain detailed regulation documents that are confidential to Consolidated Edison Company of New York Inc. (Con Edison) but are available from the corresponding author on reasonable request.

Declarations

Conflict of interest No potential conflict of interest was reported by the authors.

References

1. Russell S, Norvig P (2002) Artificial intelligence: a modern approach
2. Parunak HVD (1996) Applications of distributed artificial intelligence in industry. Found Distrib Artif Intell 2(1):18
3. House W (2016) Artificial intelligence, automation, and the economy. Executive office of the President. https://obamawhitehouse.archives.gov/sites/whitehouse. gov/files/documents/Artificial-Intelligence-Automation-Economy. PDF
4. Lee J, Davari H, Singh J, Pandhare V (2018) Industrial artificial intelligence for industry 4.0-based manufacturing systems. Manuf Lett 18:20–23
5. Ramesh A, Kambhampati C, Monson JR, Drew P (2004) Artificial intelligence in medicine. Ann R Coll Surg Engl 86(5):334

6. Hamet P, Tremblay J (2017) Artificial intelligence in medicine. Metabolism 69:36–40
7. He J, Baxter SL, Xu J, Xu J, Zhou X, Zhang K (2019) The practical implementation of artificial intelligence technologies in medicine. Nat Med 25(1):30–36
8. Taylor RH, Menciassi A, Fichtinger G, Fiorini P, Dario P (2016) Medical robotics and computer-integrated surgery. In: Springer Handbook of Robotics, pp. 1657–1684. Springer, Cham
9. Kononenko I (2001) Machine learning for medical diagnosis: history, state of the art and perspective. Artif Intell Med 23(1):89–109
10. Bakator M, Radosav D (2018) Deep learning and medical diagnosis: a review of literature. Multimodal Technol Interact 2(3):47
11. Bland M (2015) An introduction to medical statistics. Oxford University Press, UK
12. Tarca AL, Carey VJ, Chen X-W, Romero R, Drăghici S (2007) Machine learning and its applications to biology. PLoS Comput Biol 3(6):116
13. Bahrammirzaee A (2010) A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. Neural Comput Appl 19(8):1165–1195
14. Lv D, Yuan S, Li M, Xiang Y (2019) An empirical study of machine learning algorithms for stock daily trading strategy. Math Probl Eng 2019
15. Arévalo A, Niño J, Hernández G, Sandoval J (2016) High-frequency trading strategy based on deep neural networks. In: International Conference on Intelligent Computing, pp. 424–436. Springer
16. Mishra M, Nayak J, Naik B, Abraham A (2020) Deep learning in electrical utility industry: a comprehensive review of a decade of research. Eng Appl Artif Intell 96:104000
17. Goralski MA, Tan TK (2020) Artificial intelligence and sustainable development. Int J Manag Educ 18(1):100330
18. Hasan K, Shetty S, Ullah S (2019) Artificial intelligence empowered cyber threat detection and protection for power utilities. In: 2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC), pp. 354–359. IEEE
19. Momoh JA, El-Hawary ME (2018) Electric systems, dynamics, and stability with artificial intelligence applications. CRC Press, USA
20. Desatnick RL (1987) Managing to keep the customer: how to achieve and maintain superior customer service throughout the organization. Jossey-Bass, USA
21. Lee SM, Lee D (2020) untact: a new customer service strategy in the digital age. Serv Bus 14(1):1–22
22. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
23. Rish I et al (2001) An empirical study of the naive bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence 3: 41–46
24. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
25. Haykin S (2009) Neural networks and learning machines. 3/EPearson Education India, India
26. Tang Y, Jin B, Sun Y, Zhang Y-Q (2004) Granular support vector machines for medical binary classification problems. In: 2004 Symposium on Computational Intelligence in Bioinformatics and Computational Biology, pp. 73–78. IEEE
27. Villalba-Diez J, Schmidt D, Gevers R, Ordieres-Meré J, Buchwitz M, Wellbrock W (2019) Deep learning for industrial computer vision quality control in the printing industry 4.0. Sensors 19(18):3987
28. Xu J, Li H (2007) Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 391–398
29. Nallapati R (2004) Discriminative models for information retrieval. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 64–71
30. Safavian SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21(3):660–674
31. Schapire RE (2013) Explaining adaboost. Empirical Inference, pp. 37–52. Springer, Berlin
32. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, pp. 785–794
33. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y (2017) Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Inform Proce Syst 30
34. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
35. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386
36. Gevrey M, Dimopoulos I, Lek S (2003) Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecol Model 160(3):249–264
37. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H (2018) State-of-the-art in artificial neural network applications: A survey. Heliyon 4(11):00938
38. LeCun Y, Bengio Y et al (1995) Convolutional networks for images, speech, and time series. Handb Brain Theory Neural Netw 3361(10):1995
39. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inform Proc Syst 25
40. Nassif AB, Shahin I, Attili I, Azzeh M, Shaalan K (2019) Speech recognition using deep neural networks: a systematic review. IEEE Access 7:19143–19165
41. Feichtenhofer C, Fan H, Malik J, He K (2019) Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6202–6211
42. Iqbal T, Qureshi S (2020) The survey: Text generation models in deep learning. J King Saud Univ Comput Inform Sci
43. Medsker LR, Jain L (2001) Recurrent neural networks. Des Appl 5:64–67
44. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
45. Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259
46. Aly M (2005) Survey on multiclass classification methods. Neural Netw 19(1–9):2
47. De Boer P-T, Kroese DP, Mannor S, Rubinstein RY (2005) A tutorial on the cross-entropy method. Ann Oper Res 134(1):19–67
48. Tsoumakas G, Katakis I (2007) Multi-label classification: an overview. Int J Data Warehous Min (IJDWM) 3(3):1–13
49. Tarekegn AN, Giacobini M, Michalak K (2021) A review of methods for imbalanced multi-label classification. Pattern Recogn 118:107965
50. Gunasekara I, Nejadgholi I (2018) A review of standard text classification practices for multi-label toxicity identification of online content. In: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), pp. 21–25
51. Luo X, Zincir-Heywood AN (2005) Evaluation of two systems on multi-class multi-label document classification. In:

International Symposium on Methodologies for Intelligent Systems, pp. 161–169. Springer
52. Cerri R, Barros RC, de Carvalho PLF AC, Jin Y (2016) Reduction strategies for hierarchical multi-label classification in protein function prediction. BMC Bioinform 17(1):1–24
53. Li T, Ogihara M, Li Q (2003) A comparative study on content-based music genre classification. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pp. 282–289
54. Trohidis K, Tsoumakas G, Kalliris G, Vlahavas IP et al (2008) Multi-label classification of music into emotions. ISMIR 8:325–330
55. Oramas S, Nieto O, Barbieri F, Serra X (2017) Multi-label music genre classification from audio, text, and images using deep features. arXiv preprint arXiv:1707.04916
56. Boutell MR, Luo J, Shen X, Brown CM (2004) Learning multi-label scene classification. Pattern Recogn 37(9):1757–1771
57. Zhang Z-L, Zhang M-L (2006) Multi-instance multi-label learning with application to scene classification. Adv Neural Inform Process Syst 19
58. Qi X, Zhu P, Wang Y, Zhang L, Peng J, Wu M, Chen J, Zhao X, Zang N, Mathiopoulos PT (2020) Mlrsnet: a multi-label high spatial resolution remote sensing dataset for semantic scene understanding. ISPRS J Photogramm Remote Sens 169:337–350
59. Cherman EA, Monard MC, Metz J (2011) Multi-label problem transformation methods: a case study. CLEI Electron J 14(1):4–4
60. Spolaôr N, Cherman EA, Monard MC, Lee HD (2013) A comparison of multi-label feature selection methods using the problem transformation approach. Electron Notes Theor Comput Sci 292:135–151
61. Read J (2008) A pruned problem transformation method for multi-label classification. In: Proc 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008), vol. 143150, p. 41
62. Prajapati P, Thakkar A, Ganatra A (2012) A survey and current research challenges in multi-label classification methods. Int J Soft Comput Eng (IJSCE) 2(1):248–252
63. Santos A, Canuto A, Neto AF (2011) A comparative analysis of classification methods to multi-label tasks in different application domains. Int. J. Comput. Inform. Syst. Indust. Manag. Appl 3:218–227
64. Ben-Baruch E, Ridnik T, Zamir N, Noy A, Friedman I, Protter M, Zelnik-Manor L (2020) Asymmetric loss for multi-label classification. arXiv preprint arXiv:2009.14119
65. Davidson J, Liebald B, Liu J, Nandy P, Van Vleet T, Gargi U, Gupta S, He Y, Lambert M, Livingston B, et al (2010) The youtube video recommendation system. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 293–296
66. Jain H, Prabhu Y, Varma M (2016) Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–944
67. Kumar P, Thakur RS (2018) Recommendation system techniques and related issues: a survey. Int J Inf Technol 10(4):495–501
68. Chalkidis I, Fergadiotis M, Kotitsas S, Malakasiotis P, Aletras N, Androutsopoulos I (2020) An empirical study on large-scale multi-label text classification including few and zero-shot labels. arXiv preprint arXiv:2010.01653
69. Zhang Y, Wang Y, Liu X-Y, Mi S, Zhang M-L (2020) Large-scale multi-label classification using unknown streaming images. Pattern Recogn 99:107100
70. Zhang M-L, Li Y-K, Liu X-Y, Geng X (2018) Binary relevance for multi-label learning: an overview. Front Comp Sci 12(2):191–202
71. Babbar R, Schölkopf B (2017) Dismec: Distributed sparse machines for extreme multi-label classification. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 721–729
72. Babbar R, Schölkopf B (2019) Data scarcity, robustness and extreme multi-label classification. Mach Learn 108(8):1329–1351
73. Prabhu Y, Varma M (2014) Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 263–272
74. Liu J, Chang W-C, Wu Y, Yang Y (2017) Deep learning for extreme multi-label text classification. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 115–124
75. Zhang W, Yan J, Wang X, Zha H (2018) Deep extreme multi-label learning. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, pp. 100–107
76. You R, Zhang Z, Wang Z, Dai S, Mamitsuka H, Zhu S (2019) Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. Adv Neural Inform Process Syst 32
77. Bhatia K, Jain H, Kar P, Varma M, Jain P (2015) Sparse local embeddings for extreme multi-label classification. Adv Neural Inform Process Syst 28
78. Jalan A, Kar P (2019) Accelerating extreme classification via adaptive feature agglomeration. arXiv preprint arXiv:1905.11769
79. Evans N, Levinson SC (2009) The myth of language universals: language diversity and its importance for cognitive science. Behav Brain Sci 32(5):429–448
80. Black M (2019) The importance of language. Cornell University Press
81. Anderson SR (2010) How many languages are there in the world. Linguistic Society of America
82. Nadkarni PM, Ohno-Machado L, Chapman WW (2011) Natural language processing: an introduction. J Am Med Inform Assoc 18(5):544–551
83. Hirschberg J, Manning CD (2015) Advances in natural language processing. Science 349(6245):261–266
84. Chowdhary K (2020) Natural language processing. Fundam Artif Intell 603–649
85. Yadav A, Vishwakarma DK (2020) Sentiment analysis using deep learning architectures: a review. Artif Intell Rev 53(6):4335–4385
86. Behera B, Kumaravelan G, Kumar P (2019) Performance evaluation of deep learning algorithms in biomedical document classification. In: 2019 11th International Conference on Advanced Computing (ICoAC), pp. 220–224. IEEE
87. Rahman S, Chakraborty P (2021) Bangla document classification using deep recurrent neural network with bilstm. In: Proceedings of International Conference on Machine Intelligence and Data Science Applications, pp. 507–519. Springer
88. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2019) Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461
89. Hashimoto TB, Zhang H, Liang P (2019) Unifying human and statistical evaluation for natural language generation. arXiv preprint arXiv:1904.02792
90. Anandarajan M, Hill C, Nolan T (2019) Text preprocessing. In: Practical Text Analytics, pp. 45–59. Springer, Cham

91. Zhang Y, Jin R, Zhou Z-H (2010) Understanding bag-of-words model: a statistical framework. Int J Mach Learn Cybern 1(1):43–52
92. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
93. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar. https://doi.org/10.3115/v1/D14-1181. https://aclanthology.org/D14-1181
94. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Netw 18(5–6):602–610
95. Huang Z, Xu W, Yu K (2015) Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991
96. Hu D (2019) An introductory survey on attention mechanisms in nlp problems. In: Proceedings of SAI Intelligent Systems Conference, pp. 432–448 . Springer
97. McCann B, Bradbury J, Xiong, C., Socher, R (2017) Learned in translation: Contextualized word vectors. Adv Neural Inform Process Syst 30
98. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana. https://doi.org/10.18653/v1/N18-1202. https://aclanthology.org/N18-1202
99. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
100. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
101. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inform Process Syst 30
102. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692
103. He P, Liu X, Gao J, Chen W (2020) Deberta: Decoding-enhanced bert with disentangled attention. arXiv preprint arXiv:2006.03654
104. Pappagari R, Zelasko P, Villalba J, Carmiel Y, Dehak N (2019) Hierarchical transformers for long document classification. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 838–844. IEEE
105. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489
106. Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res 21(1):5485–5551
107. Roberts A, Chung HW, Levskaya A, Mishra G, Bradbury J, Andor D, Narang S, Lester B, Gaffney C, Mohiuddin A et al (2022) Scaling up models and data with t5x and seqio. arXiv preprint arXiv:2203.17189**13**
108. Xue L, Barua A, Constant N, Al-Rfou R, Narang S, Kale M, Roberts A, Raffel C (2022) Byt5: towards a token-free future with pre-trained byte-to-byte models. Trans Assoc Compu Linguist 10:291–306
109. Zhuang H, Qin Z, Jagerman R, Hui K, Ma J, Lu J, Ni J, Wang X, Bendersky M (2022) Rankt5: Fine-tuning t5 for text ranking with ranking losses. arXiv preprint arXiv:2210.10634
110. Yu C, Shen Y, Mao Y (2022) Constrained sequence-to-tree generation for hierarchical text classification. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1865–1869
111. Chen X, Xu J, Wang A (2020) Label representations in modeling classification as text generation. In: Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pp. 160–164. Association for Computational Linguistics, Suzhou, China . https://aclanthology.org/2020.aacl-srw.23
112. Qin C, Joty S. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In: International Conference on Learning Representations
113. Lester B, Al-Rfou R, Constant N (2021) The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3045–3059
114. He Y, Zheng S, Tay Y, Gupta J, Du Y, Aribandi V, Zhao Z, Li Y, Chen Z, Metzler D, et al (2022)Hyperprompt: Prompt-based task-conditioning of transformers. In: International Conference on Machine Learning, pp. 8678–8690. PMLR
115. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
116. Devlin J, Chang M, Lee K, Toutanova K (2018) BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805**arxiv:1810.04805
117. Smagulova K, James A (2019) A survey on lstm memristive neural network architectures and applications. Eur Phys J Spec Top. https://doi.org/10.1140/epjst/e2019-900046-x
118. Adhikari A, Ram A, Tang R, Lin J (2019) Rethinking complex neural network architectures for document classification. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4046–4051. Association for Computational Linguistics, Minneapolis, Minnesota. https://doi.org/10.18653/v1/N19-1408.https://aclanthology.org/N19-1408
119. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
120. Adhikari A, Ram A, Tang R, Lin J (2019) Docbert: BERT for document classification. CoRR **abs/1904.08398**arxiv:1904.08398

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Tolga Dimlioglu[1] · Jing Wang[1] · Devansh Bisla[1] · Anna Choromanska[1] · Simon Odie[2] · Leon Bukhman[2] · Afolabi Olomola[2] · James D. Wong[2]

✉ Jing Wang
jw5665@nyu.edu

Tolga Dimlioglu
td2249@nyu.edu

Devansh Bisla
bisla@nyu.edu

Anna Choromanska
ac5455@nyu.edu

Simon Odie
ODIES@coned.com

Leon Bukhman
BukhmanL@coned.com

Afolabi Olomola
OLOMOLAA@coned.com

James D. Wong
WONGJA@coned.com

[1] Department of Electrical and Computer Engineering, New York University, 5 Metro Tech, Brooklyn, NY 11201, USA

[2] Research and Development Department, et al., Consolidated Edison Company of New York Inc. (Con Edison), 4 Irving Place, New York, NY 10003, USA