Data-Centric Content Classification of Smart City Residential Services

Ding Li \(^1\), Jincheng Sun \(^2\), Yan Liu \(^2\), and Wenjie Du \(^2\)

\(^1\)Concordia University
\(^2\)Affiliation not available

October 30, 2023

Abstract

The residential services in the context of smart cities accumulate massive real-time inquiry data in natural language to describe the services in need. Such inquiry requests have diverse topics in the content and considerable variate length. Besides, the responsible departments that may handle the inquiry involve a large number of organizations, from metropolitan administration to local communities. Hence accumulated request data is primary and central to the service of accurately dispatching requests to responsible departments. The challenge is devising a data centric approach to fit the data with SOTA models and improve the request classification accuracy. In this paper, we analyze the factors of embedding tokens, data segmentation, model structures, and classification methods. We devise a unified modelling process with multiple dataflows that combine these factors to observe their interactions. The experiment results demonstrate the compound effects and provide insights into how SOTA models respond differently to variations in these factors. The observations allow us to fine-tune the learning task at each stage and achieve a maximum 82.4\% F1-Score.
Data-Centric Content Classification of Smart City Residential Services

Ding Li\textsuperscript{1}, Jincheng Sun, Yan Liu\textsuperscript{2}, and Wenjie Du

Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, yan.liu@concordia.ca

Abstract. The residential services in the context of smart cities accumulate massive real-time inquiry data in natural language to describe the services in need. Such inquiry requests have diverse topics in the content and considerable variate length. Besides, the responsible departments that may handle the inquiry involve a large number of organizations, from metropolitan administration to local communities. Hence accumulated request data is primary and central to the service of accurately dispatching requests to responsible departments. The challenge is devising a data centric approach to fit the data with SOTA models and improve the request classification accuracy. In this paper, we analyze the factors of embedding tokens, data segmentation, model structures, and classification methods. We devise a unified modelling process with multiple dataflows that combine these factors to observe their interactions. The experiment results demonstrate the compound effects and provide insights into how SOTA models respond differently to variations in these factors. The observations allow us to fine-tune the learning task at each stage and achieve a maximum 82.4\% F1-Score.

Keywords: Data Centric Deep Learning, Content Classification, Natural Language Processing, Smart City

1 Introduction

In the trend of smart city development, the ability to direct the local residential requests to corresponding metropolitan administration or service departments is necessary for improving the efficiency of administration, services and life quality. Currently, operators in the call centers receive phone calls and emails in natural language and dispatch these requests from residents to a responsible department to handle each case. Automating this process involves the task of NLP-based content classification from the input of the residential request data to the output as the corresponding department name.

The input data is request descriptions about the problem to be addressed, specific report, demand or need. The length of the description varies, ranging from containing only one sentence to multiple paragraphs. Moreover, the semantics involved in the content is diverse. Some descriptions directly present keywords and names to map with the target department, while others may contain redundant information and lack explicit clues even for human operators. The output data usually contains the same departments in facts with subtle acronyms or ambiguous terms. Furthermore, request distribution to the target departments is unbalanced with data skewness. In more detail, 20\% labels
(fourteen out of seventy-three departments) take over 76.9% requests (490,533 out of 637,917 requests). The characters of the data lead to two issues for the classification problem, namely 1) how to produce the labels of the responsible departments for the purpose of classification; and 2) What are the techniques to handle variable length and rich-semantic of the request descriptions as features for certain NLP deep learning models?

Our previous work [1] addressed the first issue by adopting means of unsupervised clustering algorithms to generate meta-class labels. For the second issue, our previous work developed a Word2Vec [2] model as word embedding and a Residual Neural Network [1, 3] as the classifier model. The limitation is that Word2Vec embedding tends to measure the similarity between words. Words with similar meanings are grouped in the hidden space, neglecting the relationship between words with different meanings [2]. Residual Neural Network, which is a convolution-based structure, lacks the ability to distinguish between words with highly distinct meanings but similar spellings [4]. Thus, our previous has limitations in catching the comprehensive semantics.

In this paper, we build pre-trained transformer-based models with the corpus created from whole request samples. The pre-trained transform models produce the word sequences relevant to the responsible department. We propose a Sequence-to-sequence (Seq2seq) structure-based word matching workflow to match the meta-class labels of responding departments. This workflow is considered to bring performance improvement. We develop data-centric experiments to observe the interaction between model performance and data-related factors. We focus on the following research questions:

**RQ1:** What is the effect of word embedding techniques combined with different classifiers?

**RQ2:** Does the length of data segmentation affect the learning performance?

**RQ3:** Comparing with the direct classification, is there performance improvement using word matching?

The main contributions of our work are summarized as follows:

– We develop a word matching method by extending the transformer model to generate the responsible department names with the highest probability of matching the class labels. The performance of this approach achieve a maximum 82.4% F1-Score with marginal improvement than direct classification.

– We design data-centric process, dataflows and experiments to observe the compound effects from multiple sources including word embedding, model structures, classification methods, and data segmentation length that increasing the length of the input sequence does not have significant efforts on performance in our dataset.

– The model implementation and pre-processed dataset discussed in this paper are open-sourced and available in github[^1].

[^1]: https://github.com/DataCentricClassificationofSmartCity/DataCentric

The structure of the paper is organized as follows: section 2 introduces the related work. The dataset description and the feature engineering methods are
-described in section 3. Section 4 discusses two methods of direct classification and sequence matching of the responsible department. Section 5 outlines the classification models adopting state-of-the-art models. Section 6 presents the experiments and observations to research questions. Threats to validity are listed in section 7 and the paper concludes in section 8.

2 Related Work

Text Classification. In recent years, deep learning approaches have achieved the current state-of-the-art performance on text classification tasks. Convolutional Neural Networks (CNN) models have been proved able to perform well on sentence-level classification task [5]. Kim trains a simple CNN with only one convolutional layer on word vectors produced from an unsupervised Neural Network Language Model (NNLM). It handles sentence classification problems quite well, which reveals the potential of CNNs on natural selection language processing tasks. At the same time, they are combined with pre-trained word vectors. Stuskever et al. [6] firstly proposed a Sequence-to-Sequence learning method based on the Encoder-Decoder architecture consisting of two Long Short-Term Memory (LSTM). Gehring et al. [7] achieved state-of-the-art with their Sequence-to-Sequence model that entirely relied on convolution. However, Bahdanau et al. argued that such architecture with a fixed-length vector was hindered from better performance, mainly because the fixed-length made long sentences quite challenging for the model to deal with [8]. Moreover, their proposed architecture enables models to select a set of words and learn to align them with target words, which now is called the attention mechanism. In 2017, Transformer with a self-attention mechanism was proposed and changed the game in NLP field [9]. After, more and more models derived from or learning ideas from Transformer. For instance, BERT [10], XLNet [11] are achieving increasingly better results not only in classification tasks but a broad set of tasks in the NLP field.

Transfer Learning Approaches In Text Classification. Transfer learning, whose importance has been demonstrated in computer vision (CV) [12], has played an increasingly critical role in NLP recently as well. In the NLP field, applications of transfer learning often pre-training models with unsupervised methods on large unlabeled datasets so that it can help models obtain general knowledge to understand text [13]. Moreover, in the fine-tune period, this kind of knowledge can be transferred to models on downstream tasks, which can not only alleviate the dilemma of small datasets available on specific downstream tasks but also lower the barrier in NLP because pre-trained models can be recycled that makes fine-tuning straightforward and much less expensive. This pre-trained fine-tune and attention-based approach has achieved many state-of-the-art models in recent years and has been popular in the NLP domain [10, 11]. XLNet [11], a recently proposed permutation-based autoregressive (AR) pre-training model, outperforms BERT, which was the SOTA pre-training method, on a wide range of NLP tasks. In discussing what results in XLNet’s success, Yang et al. [11] credit it to two attributes of XLNet. One is a permutation that enables XLNet to model deep bidirectional contexts. The other is that as an AR
language model, XLNet is born to model the joint probability with the product rule, which is that BERT cannot. Moreover, XLNet does not need to be trained with masked sequences, which is one objective of BERT pre-training. So that XLNet avoids suffering from the discrepancy between pre-train and fine-tune for the mask symbol only exists in pre-training data but not in fine-tune’s.

3 Feature Engineering

The task of feature engineering includes tokenization, segmentation, and vectorization of the requests’ description and generating the meta-class labels of the responsible departments.

3.1 Data Prepossessing

The dataset contains 637,917 valid records, including full residential request descriptions and responsible department descriptions. Each sentence of the description is first split into tokens, then processed by the LTP [14] tool to remove tokens, including numbers, symbols, punctuation, and stop words. Verbs, adjectives, and adverbs are also eliminated. Organization and location-relevant nouns are combined. Tokens are now embedded in dimension vectors. We adopt Word2Vec [2] and fastText [15], two static word embedding algorithms that feed n-grams(sub-words) into the neural networks.

3.2 Pre-training Embedding Models

The corpus created from the whole dataset is used to train an embedding model to transform the token sequences into feature vectors. Figure 1 depicts a dataflow example of word embedding. We have pre-trained four models, namely Word2Vec [2], fastText [15], BERT [10] and XLNet [11]. In section 6.2, we evaluate four embedding models and compare a model’s effect on learning accuracy.

3.3 Processing Classification Labels

The classification labels refer to the meta-classes of responsible departments. The raw data contains more than three hundred and sixty responsible departments. These descriptions usually contain the location phrases of different administrative levels, such as district, city, province etc. Many department descriptions share similar semantics that refers to a single department name. In our previous work [1], a manually created dictionary mapped the same departments described by different location phrases to the same label, resulting in seventy-three classes.

3.4 Data Sequence Segmentation

In a request description, the major claim or description mainly occurs in the first half of the description. Hence we assume that the responsible department should...
be inferred based on the early part of the request description. For example, we can reason the following request in Figure 2 should be dispatched to "Municipal security and administration" from "Street lights are not on". To observe the interaction between the sequence length and inference accuracy, we perform a set of experiments by various sequence segmentation lengths.

![Fig. 2: Example of the emphasis of a request description](image)

4 Direct Classification and Word Matching

The learning goal is mapping individual request descriptions to one of the seventy-three class labels. The class labels indicate the responsible department without location information, reducing the number of class labels. Location information can be later composed of the class label. We develop two methods, one is direct classification, and the other is matching the sequences of the model output and the class labels in tokens [6].

![The dataflow of direct classification and word matching](image)

(a) The dataflow of direct classification and word matching. The data flow for direct classification and word matching is depicted in Figure 3a. The original dataset is initially divided into the training set and testing set by the proportion of eighty percentage and twenty percentage. The dataset is first processed to generate the classification labels as described in section 3.3. A word embedding model is pretrained based on our whole dataset as the corpus as discussed in section 3.2. The output from the word embedding model transforms the tokens into two sets of feature vectors, embedding tokens for responsible departments and the residential requests, respectively. Depending on the learning methods, the feature vectors are input to either word matching or direct classification. Both methods produce a responsible department by learning. The learning accuracy is evaluated by the same metrics.

**The Direct Classification Method.** The direct classification method produces the multi-class classification as one of the seventy-three classes. The input is the residential request feature vectors. We apply four state-of-the-art models to evaluate the classification effects, using ResNet [5], base Transformer [9], enlarged Transformer [9], and XLNet [11]. Finally, each model is stacked with a fully connected layer to output one unique class predicted. For example, the input token sequence *I lost my bike* is first embedded to become feature vectors.
Through the chain of the dataflow, the classification result is police station as shown in Figure 3b. For transformer-based models, we apply a base transformer structure and introduce the variation with cross-layer parameter sharing that enlarges the number of layers in the transformer structure. More details are presented in section 5. For the XLNet model, we pre-train the model with the whole dataset as the corpus.

Word Matching. The method of word matching adopts the encoder-decoder structure, following two stages. In the first stage, the encoder-decoder transformer model is adopted to generate the word sequence related to the responsible departments. As displayed in Figure 4a, the encoder takes the input of feature vectors of residential requests. The decoder has two sources of inputs. One is the output from the residential request encoding sequence. Likewise, the responsible department available as the data label from the original dataset is embedded and fed into the decoder structure but placed in the start token <s>.

The output from the decoder is input to the fully connected layer to produce the predicted sequence of the responsible department. Lacking encoder-decoder structure, Residual CNN adopts a multi-label classification strategy to perform this experiment to mimic Word Matching based on Transformer structure. The natural language of the responsible departments is multi-labelled word by word. The results of the multi-label classification are grouped into the sequence for string matching by FuzzyWuzzy.

In the second stage, the sequence is then matched with the seventy-three class labels as shown in Figure 4b. The matching utilizes a fuzzy matching technique [16] that uses Levenshtein Distance to calculate the differences between sequences and generates the similarity score. The Levenshtein Distance leverages the number of transformations (deletions, insertions, or substitutions) required to convert a text sequence into the target one. We select the final result using the class label with the highest matching score. As Figure 4b shows, the input request description I lost my bike is ultimately predicted to the words sequence police station <eos>, where <eos> is the end-of-sentence token.
5 The Classification Models

We adopt state-of-the-art deep learning models for the content classification, including the Residual Convolutional Neural Network [3] model, self-attention-based Transformer models [9, 17], and generalized auto-regressive model XLNet [11]. This section presents these models' structure, parameter settings, and how they fit our classification and word matching tasks.

5.1 The Residual Neural Network (ResNet)

We choose to use the ResNet model for the purpose of demonstrating the improvement of learning performance using our data-centric approach. ResNet was applied and demonstrated the best performance in the context of our previous work [1]. The feature vectors are embedded using the Word2Vec model. Basically, the first 100 words in a sentence are segmented to form the input and then concatenated these words as a feature map $S \in \mathbb{R}^{100 \times 100}$. The ResNet model has nineteen residual convolutional neural layers and connects to a two-layer feed-forward network to learn a multi-class classification task. Here the cross-entropy is chosen as the loss function.

5.2 The Transformer Model

In this paper, we adopt the state-of-the-art transformer model. We then introduce variation to the transformer model.

(a) A base transformer structure with three encoder-decoder layers

(b) The structure of the enlarged transformer with six encoder-decoder layers

Fig. 5: Two transformer models structures

The transformer has an encoder-decoder model structure. The encoder takes a sequence as inputs and outputs the same length $S$ sequence. Each encoder block contains a multi-head attention layer and a fully connected feed-forward layer. In multi-head attention layer, feature vectors are mapped to queries $Q$, keys $K$ and values $V$. Each vector maps an attention value by dot-production between queries and keys. The attention matrix of full feature vectors is applied to the values to assemble a new context. $H$ attention heads are generated to learn in a multi-aspect way, and then the values from different heads are aggregated by a feed-forward layer with an $(H \times E) \times E$ weight matrix. The fully connected feed-forward layer aggregates the encoder output. Each feature vector in the
sequence is processed through a max-pooling layer. A special classification token such as $< cls >$ is also added at the beginning of an input sequence to raise the model learning ability to a whole sequence representation. In practice, multiple encoder layers are stacked to learn a high-level representation.

The responsible departments are input to the decoder in a format of feature vectors for both word matching and direct classification tasks in Figure 3a. The decoder contains a masked multi-head attention layer to calculate the attention matrix over the output from the encoder. A fully connected feed-forward layer is connected after with a linear and softmax for the final output probabilities. In our implementation, we choose the embedding length for queries (Q), keys (K) and values (V) the same as the setting of sequence length in training and testing phases. The multi-attention heads number is eight as a default value in [9].

**The base transformer structure.** The base transformer model structure [9] is displayed as Figure 5a. The feature tokens from the word embedding layer are fed into the encoder block. The encoder learns the representation of the word sequence and inputs it into the decoder. The class label or sequence of the responsible department is embedded and input into the decoder for attention represented. The final probabilities of classification are input after a linear and softmax layer. In the base transformer, both three encoder and decoder blocks are stacked.

**The enlarged transformer structure.** The enlarged transformer adopts the strategy of cross-layer parameter sharing following the shared-attention in ALBERT [18]. The architecture of the enlarged transformer is shown in Figure 5b. Compared with the base transformer that has the three encoder and decoder layers, the enlarged transformer has six encoder and decoder layers divided into three groups. Each group has two layers forming a sharing parameters group. A variation of the base transformer model is obtained with deeper layers of representation.

5.3 Pre-training the XLNet Model

We use our whole dataset as the corpus to pre-train the XLNet model in bidirectional context. This pre-trained model is then used in the direct classification method as presented in Figure 3b. The XLNet model adopts the Permutation Language Modeling (PLM) objective for auto-regressive learning. By altering the order of the input sequence in training, the model not only captures information from left to right context direction but also from right to left.

In the architecture, the XLNet model utilizes a two-stream self-attention structure for target-aware representations. It has two types of self-attention, including content and query. The content representation is similar to the standard self-attention in the transformer model that contains both content and position information. The query representation only considers the target token’s position information and the context before the current token’s position.

**Pre-training the XLNet model.** Before the downstream task, a $< cls >$ token is added at the beginning of every sequence. The input sequences are first factorized into lists of order-factorized sequences by PLM. Next, the XLNet model anticipates the subsequent words based on the preceding words [11]. We
apply a twelve two-stream self-attention layers architecture with seven hundred and sixty-eight hidden states and twelve attention heads. This XLNet model is then trained by corpus derived from the whole dataset after the data processing described in section 3.1.

**Fine-tuning the XLNet model.** The XLNet pre-trained model is usually fine-tuned with an extra linear layer or multi-layer perceptron (MLP) to perform an application specific classification task. We adopt a regularization technique, called Label Smoothing [11] that introduces noise for the labels. Label Smoothing regularizes our XLNet model based on a softmax with $k = 73$ output values by replacing the hard 0 and 1 classification targets with targets of $\frac{1}{k} \epsilon$ and $1 - \epsilon$ respectively, where $\epsilon = 0.1$.

### 6 The Evaluation

We design experiments and carry out measurements to evaluate observations according to the four research questions proposed at the beginning of this paper. The evaluation metrics include Precision, Recall and F1-Score.

#### 6.1 Evaluating Classification Performance (RQ1)

**Objective:** Experiment One is performed to observe the comparison of classification performance among ResNet, base transformer, enlarged transformer models and the XLNet model.

Table 1: Classification performance by four machine learning models

<table>
<thead>
<tr>
<th>Models</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
</tr>
<tr>
<td>Residual CNN</td>
<td>0.788</td>
</tr>
<tr>
<td>Base Transformer</td>
<td>0.814</td>
</tr>
<tr>
<td>Enlarged Transformer</td>
<td>0.824</td>
</tr>
<tr>
<td>XLNet</td>
<td>0.829</td>
</tr>
</tbody>
</table>

*Input Sequence Length = 100.*

**Result:** In this experiment, the input sequence length is kept constant at one hundred. ResNet, base transformer and enlarged transformer models apply the same Word2Vec embedding model that is pre-trained on the whole dataset as a corpus. The XLNet adopts its own embedding model based on the same corpus. The test dataset of 127,583 samples is input as the residential requests, and the classification results are the responsible department as one of the seventy-three class labels. The classification performance of the four models is summarised in Table 1. **Conclusion:** The two transformer models outperform the ResNet in terms of precision and recall. The enlarged transformer model enhances marginal recall performance by implementing a cross-layer parameter sharing strategy and a multi-layer structure. The XLNet model outperforms other models based on all metrics. Since the XLNet applies a different word embedding method than other models, we cannot reach the conclusion the performance improvement is due to XLNet structure, the embedding or both.

#### 6.2 Effect of Embedding Models (RQ1)

**Objective:** Experiment Two further evaluates the effects of embedding models in addition to Word2Vec.
Result: Four SOTA embedding models are considered, namely Word2Vec, fastText [15], BERT [10], and XLNet. Since XLNet uses its intrinsic embedding model, we have a combination between the four embedding models and ResNet, base transformer and enlarged transformer models. The experimental results are presented in Table 2 as Precision, Recall, and F1-score in Macro. Each entry of Table 2 reads as the classification model (such as ResNet) combined with the embedding model (such as Word2Vec) produces the precision or recall or F1-score, respectively. Conclusion: For each model, the four embedding models produce comparable metrics values with Experiment one. XLNet still outperforms other combinations. For the other three classification models, the effects of different embedding models are insignificant. Word2Vec and fastText yield metrics are approximately close to each other. The BERT model improves the F1-score of the transformer models slightly. It is worth noting that XLNet as an embedding model even degrades the metrics of other classification models.

<table>
<thead>
<tr>
<th>Models</th>
<th>Embedding Methods</th>
<th>Word2Vec</th>
<th>fastText</th>
<th>BERT</th>
<th>XLNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Residual CNN</td>
<td>0.788</td>
<td>0.694</td>
<td>0.738</td>
<td>0.769</td>
<td>0.661</td>
</tr>
<tr>
<td>Base Transformer</td>
<td>0.814</td>
<td>0.744</td>
<td>0.773</td>
<td>0.814</td>
<td>0.744</td>
</tr>
<tr>
<td>Enlarged Transformer</td>
<td>0.824</td>
<td>0.749</td>
<td>0.783</td>
<td>0.776</td>
<td>0.727</td>
</tr>
<tr>
<td>XLNet</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

*Input Sequence Length = 100.*

6.3 Evaluating Data Sequence Segmentation (RQ2)

Objective: Experiment Three evaluates the effects of data sequence segmentation length. The original residential requests have varied lengths. According to the input sequence length statistics, the distribution of input lengths of twenty-five, fifty, one hundred, and two hundred are 30.3%, 48.1%, 81%, 94% respectively. We input sequences with the length of twenty-five, fifty, one hundred, and two hundred, respectively. These sequences are input to embedding models (Word2Vec for ResNet and transformer models) and the XLNet model.

<table>
<thead>
<tr>
<th>Models</th>
<th>Input Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>l = 25</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
</tr>
<tr>
<td>Residual CNN</td>
<td>0.771</td>
</tr>
<tr>
<td>Base Transformer</td>
<td>0.764</td>
</tr>
<tr>
<td>Enlarged Transformer</td>
<td>0.759</td>
</tr>
<tr>
<td>XLNet</td>
<td>0.762</td>
</tr>
</tbody>
</table>

Result: The results are shown in Table 3. When the length is one hundred and two hundred, the precision, recall and F-1 score metrics are improved compared to the length of twenty-five and fifty. This trend is across the four models. Conclusion: Varying the data segmentation length has effects on the learning metrics. Empirical experiments are helpful in setting the length of training models. Further increasing the length only gains marginal improvement in the cases of segmentation length of one hundred and two hundred.
6.4 Evaluating Classification Methods (RQ3)

Objective: Experiment Five evaluates the two classification methods, direct classification and word matching, discussed in section 4.

Result: The best classification metrics from Table 1 is extracted and placed as the result for direct classification in Table 4. Word matching results are collected by running the testing data on the word matching models as depicted in Figure 4a. Conclusion: The four models each respond differently to the classification method. The direct classification suits ResNet better than word matching does. Transformer models and XLNet has varied highlights in different metrics under either direct classification or word matching. The current experiment results indicate the choice of direct classification or word matching has marginal improvement on particular metrics given a specific model.

Table 4: Classification method comparison

<table>
<thead>
<tr>
<th>Models</th>
<th>Classification Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Direct Classification</td>
<td>0.788</td>
<td>0.694</td>
<td>0.738</td>
<td>0.781</td>
<td>0.661</td>
<td>0.716</td>
</tr>
<tr>
<td>Residual CNN</td>
<td>Word Matching</td>
<td>0.743</td>
<td>0.777</td>
<td>0.764</td>
<td>0.801</td>
<td>0.759</td>
<td>0.783</td>
</tr>
<tr>
<td>Base Transformer</td>
<td>0.749</td>
<td>0.777</td>
<td>0.764</td>
<td>0.801</td>
<td>0.759</td>
<td>0.783</td>
<td></td>
</tr>
<tr>
<td>Enlarged Transformer</td>
<td>0.732</td>
<td>0.764</td>
<td>0.759</td>
<td>0.801</td>
<td>0.759</td>
<td>0.783</td>
<td></td>
</tr>
<tr>
<td>XLNet</td>
<td>0.824</td>
<td>0.819</td>
<td>0.824</td>
<td>0.838</td>
<td>0.764</td>
<td>0.799</td>
<td></td>
</tr>
</tbody>
</table>

7 Threats To Validity

The threats to validity are from internal and external sources. The internal threat mainly comes from the dataset. The quality of the class label is not systematically evaluated. Future improvement can involve human operations to correct labels. Another threat related to the dataset is our assumption that the dataset contains typically responsible departments for metropolitan districts. In the case that a new kind of responsible department emerges in the future dataset, the classification can only classify it to the type of others. The external threat includes tuning models at the training time. The models are not fine-tuned by tools and techniques such as AutoML to optimize model hyperparameters.

8 Conclusion

In this paper, we present a data-centric development of a deep learning application in the context of the smart city. The characteristics and properties of the dataset are the drivers for devising techniques, methods and models for the whole learning process. Activities and entities in each stage of data preprocessing, word embedding, and context classification are analyzed based on the dataset. In each stage, we consider multiple techniques, methods and models and develop solutions and experiments to evaluate the effects of their interactions. The experimental results provide us with the observations of SOTA models and an understanding of compound effects from multiple factors. The improvement of learning performance is achieved through the selection of the optimal techniques and models in each combination. This data-centric approach has the contribution to demonstrate a practical approach in general to context classification applications with a large number of classification labels. The methodology is beyond the context of our demonstrated application.
References