Hybrid Machine Learning Models of Classifying Residential Requests for Smart Dispatching

Tianen Chen
ti_hen@encs.concordia.ca
Concordia University
Montreal, Quebec

Jincheng Sun
s_jinche@encs.concordia.ca
Concordia University
Montreal, Quebec

Hongyi Lin
h_ln@encs.concordia.ca
Concordia University
Montreal, Quebec

Yan Liu
yan.liu@concordia.ca
Concordia University
Montreal, Quebec

ABSTRACT

This paper presents a hybrid machine learning method of classifying residential requests in natural language to responsible departments that provide timely responses back to residents under the vision of digital government services in smart cities. Residential requests in natural language descriptions cover almost every aspect of a city's daily operation. Hence the responsible departments are fine-grained to even the level of local communities. There are no specific general categories or labels for each request sample. This causes two issues for supervised classification solutions, namely 1) the request sample data is unbalanced and 2) lack of specific labels for training. To solve these issues, our method generates meta-class labels by means of unsupervised clustering algorithms. We further develop five classifiers, including Naive Bayes, fully connected neural network, two hierarchical classifiers, and a residual convolutional neural network. Two sets of tests demonstrate our implementation of the residual convolutional neural network model produces the best classification performance on the real-world dataset. The hierarchical classification method provides insights to the source of classification errors.

KEYWORDS

machine learning, natural language processing, text classification

ACM Reference Format:
Tianen Chen, Jincheng Sun, Hongyi Lin, and Yan Liu. 2019. Hybrid Machine Learning Models of Classifying Residential Requests for Smart Dispatching . In *Proceedings of Cascon '19: INTERDISCIPLINARY. INTERCONNECTED. INTEGRATED. (Cascon'19).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The context of smart cities relates to the capability of analysing and responding to specific requests from residents. In addition to social media networks such as Twitter and Chinese Weibo, an important source of city events down to the county level are service requests directly reported from metropolitan residents. These requests are through phone calls, emails and chatbots sent to metropolitan residential service centers that employ Human agents to dispatch the requests manually to corresponding service sectors for handling such issues. However, the processing capacity of call centers is bounded by the limitation of human operation. Therefore an artificial intelligence enabled system helps to automatically convert audio-based reported requests into text content and then dispatches the request to the corresponding sector of services. Such an automation improves the service responsiveness. It also enables a scalable response to intensive request loads during special events or at seasonal periods.

The core of such an automated request analysis and dispatching system is a machine learning model that classifies a request in natural language to an organization that is responsible for handling the case. To scope the research problem, this paper has the assumption that requests are all text-based. Any audio request is converted to the text-based content. The issues of classifying these text based request data are two aspects. First, the labels for classification require processing on the raw datasets to produce suitable labels. It could be assumed that the filed called responsible department in the original dataset should be the labels. However, the responsible departments are also presented in natural language description that contains abbreviation, location information and less precise rename of a department. In addition, future requests may even result in new department names that are not part of the historical labels. Another issue is the request distribution over the corresponding responsible departments by nature are not evenly distributed. Some responsible departments are of small request cases and lead to the overall datasets unbalanced.

In this paper, we propose a hybrid method to address the above two issues. Our method consists of (1) an NLP processing workflow for feature extraction; (2) a clustering algorithm for generating meta-classes for hierarchical training; (3) multiple classifiers with a Naive Bayes model, a fully connected neural network model and a residual neural network model to produce an optimal inference for a certain request. Training the classification with generated meta-classes, we gain insights of the classification errors. A use case of our hybrid method has been developed on over 80,000 sample requests that involve 157 responsible departments. Our method

Cascon'19, November 04–06, 2019, Toronto, Ontario, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

achieves with testing precision of 76.42% and log loss value of 1.192 over 35,663 test data that collected in different time period than the training datasets.Our code is open-sourced on Github[1].

The **contribution** of this paper is three-fold as follows:

- We build an NLP-based feature engineering workflow with rigorous measure of feature prediction power using information values;
- We demonstrate a hybrid machine learning method with both unsupervised and supervised machine learning to classify residential requests over unbalanced data sample distribution;
- We develop and compare five classifiers to improve classification performance.

The structure of paper is organized as follows: Section 2 presents the related work. In Section 3, we introduce our dataset and the feature engineering techniques. Section 4 describes a hierarchical classification method of generating meta-classes for classification of large number of classes. The hybrid machine learning models in Section 5. Finally, we present our assessment metrics and experiment results of the classification performance in Section 6. We conclude our paper in Section 7.

2 RELATED WORK

Learning from few labelled data. Kamal et al. proposed an Expectation Maximization based method [21] to train a classifier with few labelled data and use the classifier to label the high-confidence unlabelled data, then use the labelled data to train a new classifier, and repeat this process until the classifier converge. Blum et al. [3] proposed a co-training method that allows learning from two views of the features. Rajat et al. proposed a Self-taught method that uses Auto-Encoder to learn higher-level representations with unlabelled data [23].

Imbalance dataset. Nitesh et al. presented a method of SMOTE (Synthetic Minority Over-sampling Technique) [5] to over-sampling the minority class by creating synthetic minority samples that fall between the minority data and their nearest neighbours. If the distance between the minority data and the neighbours is far, the synthetic data will have a large range of noise. Hui et al. presented a Borderline-SMOTE method [12] ensures that the sampling happens only when the majority of the selected neighbours are minority data to lower the randomness of noise.

Ensemble of Classifiers. Breiman proposed a bootstrap aggregating method[4] using bootstrap to extract several sub training sets from the original training set and train a classifier for each sub training set. Then each classifier gives a weighted vote for the classification. Yoav and Robert proposed a boost-based method called AdaBoost that uses weighted data to train weak classifiers. The data misclassified by a weak classifier gains more weight to train the next weak classifier. The weak classifiers are weighted to form a confident classifier. Based on AdaBoost [11] method, Wei et al. proposed a cost-sensitive Boosting [9] that increases the weight of misclassified training data that have a higher cost.

Hierarchical Classification. Pei-Yi et al proposed a hierarchically SVM text classification method that split a problem into subproblems in the classification tree that can be solved accurately and efficiently [13]. A hierarchical softmax architecture [20] was proposed by Morin and Bengio when dealing with a huge number of output classes. It significantly speeds up the training time compare to feed-forward networks.

Convolutional Neural Networks on NLP Tasks. Kim applied a single convolutional layer on top of Word2Vec embeddings [18] of words in a sentence to perform classification tasks. His work proves that convolutional neural network, with good representation of the words, can provide promising results on NLP problems.

Conneau et al. proposed ResNet-like deep convolutional neural network [7] that can learn different hierarchical representation of the input text. They use small convolutions and let the network learn what is the best combination of these extracted features. Their work allows the network to be deeper and finer-grained.

Prabowo and Thelwall proposed a series of hybrid classifiers [22] that combine different rule-based classifiers and SVM together as a pipeline to solve sentiment analysis problems and achieved good performance.

3 FEATURE ENGINEERING

Feature engineering processes and transforms the dataset in texts to word vectors as inputs of machine learning models. The original dataset contains eight features with *id*, *time stamp*, four *categories* of responsible departments, *request description*, and *responsible department description*.

The feature engineering workflow is depicted in Figure 1. Feature engineering first removes invalid data samples in which the values of the responsible department description are missing or originally marked as non-available. The training dataset contains 849,861 records, including 145,542 records originally marked as invalid and 58,394 records without a responsible department description. Finally, the valid dataset contains 645,924 records.

3.1 Data Preprocessing

Sentences are segmented into tokens before feature extraction. Two major methodologies of tokenization and segmentation are dictionary-based and statistics-based. The dictionary-based methods recognize words based on a maintained vocabulary [19], while the statistics-based approach uses corpus as a resource to build a word-segmentation model. In this paper, we process the tokenization and segmentation with the second method using a tool called LTP, a Chinese language technology platform [6]. LTP consists of six Chinese processing modules, including 1) Word Segmentation (WordSeg); 2)Part-of-Speech Tagging (POSTag); 3) Named Entity Recognition (NER); 4) Word Sense Disambiguation (WSD); 5) Syntactic Parsing (Parser) and Semantic Role Labelling (SRL).

Further more, the segmented tokens are filtered by lexical analysis modules of LTP to eliminate tokens that include digits, words in other languages, punctuation, and stopping words. A combined list of public available stopping-words is applied to the LTP tool. In addition, verbs, adjectives, adverbs are also excluded. Organization names and location relevant nouns consist of more than one tokens. The NER module of LTP recognizes and merges these nouns into single words.

[1] https://github.com/OneClickDeepLearning/classificationOfResidentialRequests

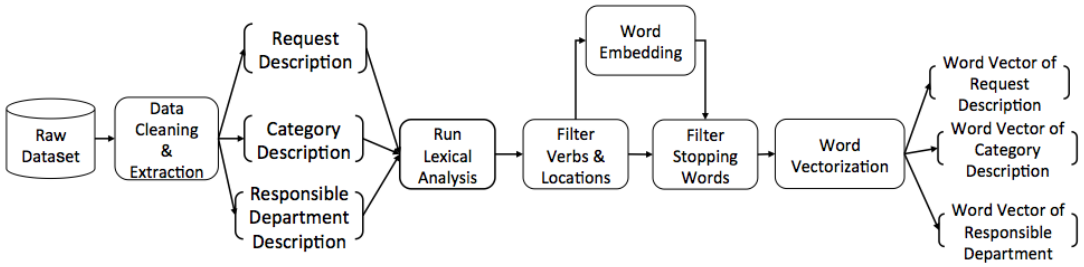

Figure 1: The major steps of feature engineering on two features of request description and responsible department description

3.2 Data Distribution

The feature that contains the labelling information is *responsible department description*. A simplified illustrating sample is depicted in Figure 2. The description needs further processing to generate labels for training and inference. The reason is the description usually contains location phrases with various levels of metropolitan granularity, national, provincial, county, and local communities. This means the records with the same responsible department but different location nouns becomes separate classes. As a result, the dataset distribution over the labels is spread widely with a large number of classes and the density of classes is diluted. Such a circumstance degrades the training quality and inference accuracy.

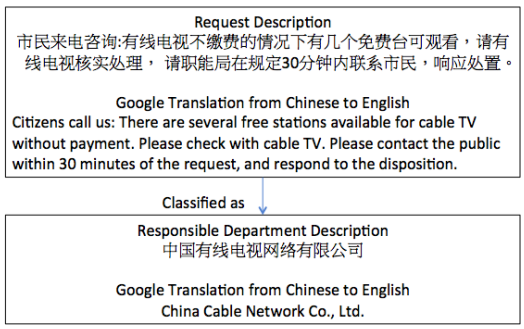

Figure 2: One data sample with features of request description and responsible department description

To solve above problem, we separate the location nouns from department names and titles. Only the organization names and titles remain to generate training labels. For cases that the department names and tiles are in abbreviation, we setup a dictionary and manually create an entry mapping between the standard full name and any forms of variation including abbreviation. This leads to 157 unique department names and titles,which are considered as the labels for classification. We further plot the data sample distributions over the 157 classes in Figure 3. Among them, there are 101 classes whose data samples are below 1000 (below the portion of 0.15%).

We further explore the dataset characteristics upon observations of data distribution. We develop an inverted index of words in each request description record. The Bag-of-Words (BoW) algorithm [14] is used to build the word pairs with their word counts per record.

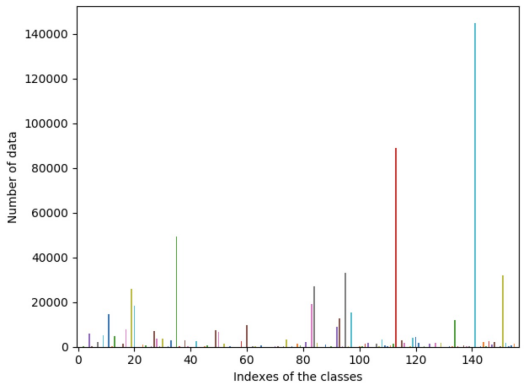

Figure 3: Data distribution statistic for 157 classes, the x-axis is the index of the classes and the y-axis is the total number of data of a class

The word and the sample become a vector that is stored. Such a vector allows us to trace the word occurrence in samples. Therefore we compute the statistics of sample counts grouped by the word occurrence frequency.

Figure 4 plots the word frequency distribution. This plot is interrupted as 9382 data samples contain words only occur once in the whole request description text; 6275 data samples contain words occur 2 to 5 times in the whole request description text. The words of high frequency (such as over 50) only appear in limited number of samples (such as 807 samples). Since stopping words are already filtered, this plot in Figure 4 indicates words with low frequency should be further measured with their relevance of other words in the feature space. Likewise, we produce the plot of word frequency and distribution of 157 labels as shown in Figure 5. Our solution is training the Word2Vec model to generate the word embedding that represent the relations of word tokens in a high dimensional space. The details are presented in Section 3.4.

3.3 Information Values of Features

Information values and Weight-of-Evidence (WoE) are techniques of feature selection. Information values measure the prediction power of a feature. The decision to select the four categories of responsible departments as features is evaluated by their information values. The value of each category are tags edited by customer

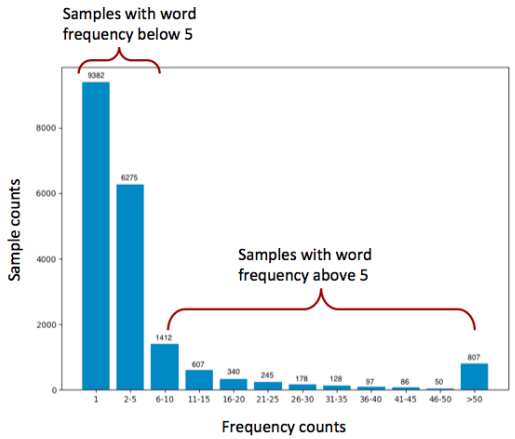

Figure 4: Data sample distribution over word frequency in request description

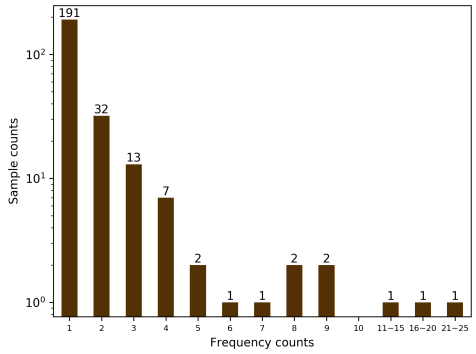

Figure 5: Label distribution over word frequency

Category 1	Category 2	Category 3	Category 4
城管问题 City Regulation	街面秩序 Street Regulation	机动车违停 Invalid Vehicle Parking	移车 Car Towelling

市公安局交通警察支队
City Police Department Transportation Station

Figure 6: Data sample of categories

service operators. A data sample is depicted in Figure 6. The bottom textbox shows the corresponding responsible department. The four categories represent four levels of tags as a whole capturing context of the request. The tokenization and segmentation workflow is applied to each category. We combine tags of four categories as a combo for the information value analysis. By statistics, there are 418 unique values of category tag combo.

$$[tag_1, tag_2, ...tag_n], n = 418$$

The information values (IVs) are calculated to measure the prediction power of the category tag combo to the class label as the responsible department. Therefore, the 157 responsible departments and the 418 category tag combo form a 157×418 vector. Each entry of this vector is notated as $TagCombo_{i,j}$, which represents the counts of data samples with tag combo j that belong to responsible department i. Then the notation of $Non - TagCombo_{i,j}$ represents the total counts of data samples with tag combo $l \neq j$, that is

$$Non - TagCombo_{i,j} = \sum_{l=1, l\neq j}^{418} \#tag - combo_{i,l}$$

Now, we can calculate the value of WoE as

$$WOE_{i,j} = ln(\frac{TagCombo_{i,j}}{Non - TagCombo_{i,j}})$$

Hence, we calculate the information value vectors for each category tag combo j for responsible department i as shown in Figure 7.

住房保障-物业服务管理-停电-故障停电
Housing security - property service management - power outage - power outage

$Responsible\ Department_i$	$TagCombo_j$	$NonTagCombo_j$	$WoE_{i,j}$	$IV_{i,j}$
供电政府服务中心 (Power Supply Service Center)	9	409	0.18598	0.01893

Figure 7: Sample Calculation of IV

Finally, the IV values of each category tag combo are summed up for all 157 responsible departments to measure the prediction power of each tag combo as

$$IV_{i,j} = (TagCombo_j\% - Non - TagCombo_j\%) * WOE_{i,j}$$

$$IV_{TagCombo_j} = \sum_{i=1}^{157} IV_{i,j}$$

According to the rule of thumb described above, we find only 13 out of 418 category-combos are weak prediction, that is IV value is in the range of (0.02,0.10]. The rest of category-combos have IV values below 0.02, which indicates not useful for prediction. In other word, it means category-combo is not an important feature for classifying responsible departments. They are not selected as input features for classifiers.

3.4 Feature Extraction

Feature extraction is to produce the word tokens into word vectors with numerical values. In this paper, we apply two methods namely Term-frequency-inverse document frequency (TF-IDF) [25],and Word2Vec [19].

3.4.1 Generating word vector using TF-IDF. TF-IDF measures the relevance of words, not frequency. That is, word counts in the inverted index discussed in Section 3.2 are replaced with TF-IDF relevance weight across the whole dataset. With TF-IDF, the more samples a word appears in, the less valuable that word is as a signal

to differentiate a given request. The relevance weight of a word is calculated in Eq 1.

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}) \quad (1)$$

where N is the number of samples; $tf_{i,j}$ represents the number of occurrence of word token i in sample request j; df_i represents the number of occurrence of request samples that contain the token i. The vector of $w_{i,j}$ is a normalized data format that adds up to one for all the samples. This TF-IDF vector is used as input to the Naive Bayes classifier discussed in Section 5.1.

3.4.2 Word embedding using Word2Vec. Word embedding maps word tokens of varied length to a fixed-length word vector as inputs to machine learning models. In nature, word embedding reconstructs linguistic contexts of words and produces a vector space. The algorithm of Word2Vec uses a group of related models that are two-layer neural networks that are trained over a large corpus of text and produces a vector space, typically of several hundred dimensions. Each unique word in the corpus is assigned a corresponding vector in the vector space so that words that share common contexts in the corpus are located in a close proximity to one another in the space [19].

We have trained the Continuous Bag-of-Words structure (CBOW) of the Word2Vec model with a corpus collected from the whole dataset. We segment the whole dataset of 65,9421 samples with sentences into data blobs of every five continuous words as input. The central word is the target for output. Empirical experiments indicate using the training dataset as corpus produces a better classification performance. The details of the experiments are not central to the research scope of this paper, thus omitted.

By statistics, we observe the word length in a request description of the dataset ranges from 1 to 780 with the weighted average of 46. Over 90% of the request descriptions have less than 100 word tokens. Thus we set the word vector dimension as 100, and pad zero if the word length is less than 100.

4 HIERARCHICAL CLASSIFICATION METHOD

A hierarchical classification method handles classification of a large number of possible classes [24]. The current training dataset contains 157 unique labels and thus 157 classes. We develop a hierarchical classification method to evaluate whether it works for our dataset. There are two kinds of hierarchical classification. One uses meta-classes as a two-hierarchy structure where leaf classes are grouped together by similarity into intermediate classes (the meta-classes) [13]. The other one copes with a pre-defined class hierarchy, a type of supervised learning. In this paper, our method is the former case by means of which we build the hierarchy during the training by a clustering method, and then classify a sample from the meta-classes to the leaf-classes.

4.1 Meta-class Generation using K-means and GMM

To create meta-classes for 157 labels according to similarity, we first apply the K-Means clustering algorithm [17]. The K-Means algorithm first assigns each sample to the cluster whose mean values have the least squared Euclidean distance. Secondly, it calculates the new means to be the centroids of the observations in the new clusters. Finally, the algorithm converges when the assignments no longer change. K-Means applies hard clustering by assigning data points to a cluster. This implies a data sample is assigned to the closest cluster. K-Means is simple to train but does not guarantee convergence to the global optimal whereby degrades the clustering accuracy.

The Gaussian Mixture Model (GMM) [2] is a finite mixture probability distribution model. The parameters of GMM are estimated iteratively by using the Expectation-Maximization (EM) algorithm [2]. GMM/EM determines a sample's probability of belonging to a cluster, which is a soft clustering process. Each cluster, therefore, can have different options to constrain the covariance such as spherical, diagonal, tied or full covariance, instead of only spherical in K-Means, which means the clustering assignment is more flexible in GMM than in K-Means. The EM algorithm has its own limitation. One issue is the number of mixtures affects the overall performance of EM and this number is an unknown prior. Therefore, the optimal number of mixtures is important to ensure an efficient and accurate estimation.

Our solution is applying K-Means to obtain values of centroids (or geometric centers), then initialize GMM with centroid values[10]. The input to K-means is a word vector with 100 dimensions of 157 class labels produced from the feature extraction process described in Section 3. We apply silhouette coefficients to select the optimal value of K. A silhouette coefficient of a range of [-1, 1] indicates (1) a sample is far away from the neighbouring clusters with the value near +1; (2)or a sample is on or very close to the decision boundary between two neighbouring clusters with the value of 0; (3) or a sample might have been assigned to the wrong cluster if the value is negative. Figure 8 plots the clustering result with the optimal K value as 5. We derive the silhouette coefficients by setting the K value range in $[3, 10]$. After the silhouette analysis, we consider $K = 5$ is the optimal value for K-Means/GMM/EM clustering. Figure 8 plots the 5 clusters of 157 labels.

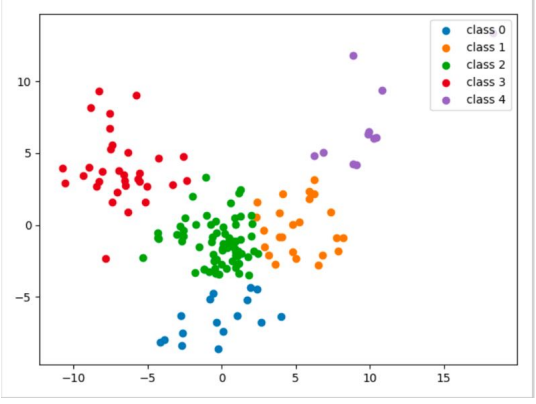

Figure 8: The plot of K-means and GMM clustering

4.2 Meta-class Generation using Topic-based Clustering

We develop another clustering method that considers the 157 labels as topics to cluster them into groups with similar themes. In this method, we apply the clustering method of OPTICS (Ordering Points To Identify the Clustering Structure Closely) [1] that finds a core sample of high density and expends clusters from them. The output from OPTICS provides K clusters. In our method, we apply LDA (Latent Dirichlet Allocation) to output K topics. Each topic consists of a fixed size of words and their associated weights. LDA assumes the Dirichlet prior distribution of topics. In practice, the Dirichlet prior distribution assumes documents (or labels in our case) cover only a small set of topics and topics consist of a small set of words frequently. Finally, we assign labels to a cluster by an entropy-based algorithm. The algorithm is listed below. The input to this topic-based algorithm is the output of the word vector representation of the 157 labels in 157×10 dimensions. The output is K clusters of 157 labels. The clustering result is shown in Figure 9.

Algorithm 1: Topics and Entropy Based Clustering

Input: $\hat{k} \leftarrow \{k_j\}, j \in [1, 157]$ word vector of 157 labels
Output: Clusters of 157 labels

$\varsigma \leftarrow$ number of clusters output from OPTICS on Input
$num\_topics \leftarrow \varsigma$ to initialize LDA;
$\hat{v_t} \leftarrow$ output from LDA; {a topic vector of $\varsigma \times 10$; each entry is a tuple t of [word: weight]}
foreach $\hat{v_t}[i], i \in [1, \varsigma]$ **do**
  $l \leftarrow |\hat{v_t}[i]|$; {number of tuples}
  $c_i = \frac{1}{|l|} \sum_{t \in \hat{v_t}[i]} t.weight$ {calculate the centroid}
  foreach $k_j, j \in [1, 157]$ **do**
    $p(k_j) = \frac{sim(k_j, c_i)}{\sum_{r=1}^{\varsigma} sim(k_j, c_r)}$
    $e_{k_j}^{c_i} = -p(k_j) \cdot log(p(k_j))$ {calculate the entropy of k_j to a topic $\hat{v_t}[i]$}
  end
end
Assign k_j to the topic cluster m whose entropy $e_{k_j}^{c_m}$ is minimal;

4.3 Hierarchical Classification

The clustering algorithms generate the meta-classes of 157 labels for training a classifier. The classification is of the structure of a two-level tree as depicted in Figure 10. The leaves are 157 labels and the non-leaf nodes are the K meta-classes.

A classifier is first trained with the meta-classes, it is noted as $Model_{meta}$. The classification decides the meta-class that a sample belongs to. As a result, the original data samples are grouped into K clusters. Furthermore, each meta-class i contains L_i labels and $\sum_{i=1}^{K} L_i = 157$. The data samples classified to one specific meta-class are used again to train a sub-model to further classify these samples to the leaf classes. The hierarchical training produces one meta-class model, noted as $Model_{meta}$ and K number of leaf-class models, one for each cluster. Totally, we have $K + 1$ models.

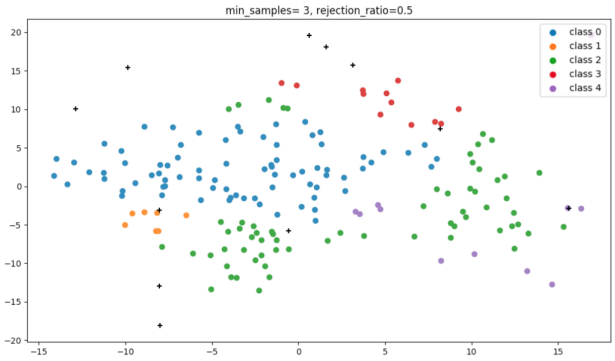

Figure 9: Topic based clustering result

When it comes to inference, there are two methods. The first method inputs the test sample to $Model_{meta}$. Based on the classification on the meta-class, the test sample is further fed to one of the leaf-class models. The second method directly inputs the test samples to each of the leaf-class models. The classification with the highest probability is selected to be the classification result. The first method has shorter inference time as it runs two models, while the second method runs K models. In term of accuracy, the second method tends to be more accurate as it reduces the error propagation from the meta-class level.

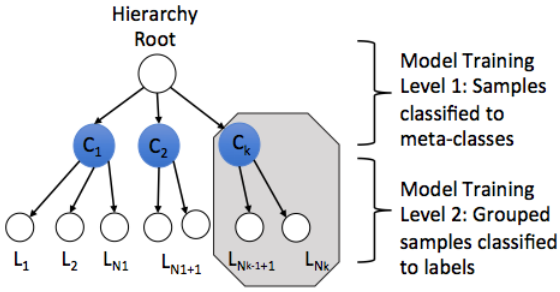

Figure 10: The two-level hierarchy of classes

This hierarchical classification method is useful to deal with a large number of classes by means of training a classifier for a much smaller number of classes (at the level of meta-classes) while keeping comparable levels of confidence. Our evaluation in section 6 shows the classification quality. Since the sample dataset is partitioned by meta-class clustering, any partition of data set is independent of each other. Thus the leaf level inference can run in parallel.

When new request samples have different labels from previous labels, the hierarchical classification method needs re-run the clustering method and then re-train the meta-class model. Then the classifier for the leaf-class model is retrained. The implementation details are beyond the scope of this paper.

The main issue with hierarchy classification is error propagation. Since the error in the meta-class level classification propagates to the leaf-class classification directly.

5 HYBRID MACHINE LEARNING MODELS

We develop two types of classifiers based on the Naive Bayes model and a fully-connected neural network model. Each model is also combined with the hierarchical classification method. In addition, we design a Residual Convolutional Neural Network model. Totally, we have developed 5 classifiers, namely (1) Naive Bayes, (2) Hierarchical Naive Bayes, (3) Fully-Connected Neural Network (FCNN), (4) Hierarchical FCNN, and (5) Residual Convolutional Neural Network (Residual CNN) classifier.

5.1 Naive Bayes Classifier

A Naive Bayes classifier is a probabilistic machine learning model based on the Bayes Theorem in Eq 2, finding the probability of y happening given X has occurred.

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)} \tag{2}$$

In the context of classification, y is replaced with a class, and X with a set of features $\{x_0, ...x_n\}$. Further, Eq 2 is transformed to Eq 3. $P(y)$ is the proportion of the dataset that belongs to this class of y.

$$P(y|x_0, ...x_n) = \frac{P(x_0, ...x_n|y) \cdot P(y)}{P(x_0, ...x_n)} \tag{3}$$

To simplify the calculation, an assumption is made as $x_0, ...x_n$ are conditionally independent given the class y, so that $P(x_0, ...x_n|y) = P(x_0|y) \times P(x_1|y) \times ...P(x_n)$. The final approximation of the class probability is given by Eq 5.

$$P(y|x_0, ...x_n) \propto P(x_0, ...x_n|y) \cdot P(y) \propto P(y) \cdot \prod_{j=1}^{n} P(x_i|y) \tag{4}$$

In the context of request classification, X contains the word tokens of the request description. Further we adopt the Bernoulli Naive Bayes classifier to encode the input of word tokens. The Bernoulli Naive Bayes classifier assumes each feature has only binary values. In this case, the word token is encoded as a one-hot bag of words model that means 1 indicating the presence of the term or 0 indicating the absence of the term. This leads to 19,607 dimensions of input features from the request description of 40,000 training set. The limitation of this approach is that when the training set changes, the input needs to be encoded again. For example, 80,000 training set leads to 32,698 dimensions of word token one-hot encoding. To avoid the zero value of $P(x_i|y_j)$ causing the posterior probability always being zero, 0.2 smooth value is added to each conditional probability $P(x_i|y_j)$. y represents a set of 157 responsible departments $\{y_1, ...y_{157}\}$. The classification is calculated as the class y_j that produces the maximum probability given the feature set of X.

$$\begin{aligned} P(\widehat{y}|x_0, ...x_n) &\propto \max_{j=1}^{157} P(y_j) \cdot \prod_{i=1}^{n} P(x_i|y_j) \\ \widehat{y} &= \arg\max_{j=1}^{157} \prod_{i=1}^{n} P(x_i|y_j) \cdot P(y_j) \end{aligned} \tag{5}$$

Hierarchical Naive Bayes. For hierarchical classification, the input remains the same. Both the meta-class model and leaf-class model use the Naive Bayes classifier. The difference is the meta-class model has 5 classes for classification.

5.2 Neural Network Classifiers

By applying a neural network model to the task of text classification, we assume the output from the complex function created using the network of neurons is close to the true relationship between the input features (such as word tokens of request description) and the output (in our case the 157 classes of responsible departments). In another words, a neural network with a certain number of hidden layers should be able to approximate any function that exits between the input and the output. The proof of Universal Approximation Theorem [8] satisfies this assumption.

We develop two types of neural network classifiers. One is based on a Fully Connected Neural Network (FCNN) model. The other is developed using the Convolutional Neural Network (CNN) model with special skip connections and use of batch normalization (ResNet). Both neural network models take the input as the Word2Vec embedding of the word tokens processed from the request description of each sample. The technical details are presented in Section 3.4.2. The input word vector is of 100×100 dimensions.

Fully Connected Neural Network. In a FCNN model, each input neuron is connected to each output neuron in the next layer, referred to as a Dense layer. Given the input to the first Dense layer is of the size of $10,000 = 100 \times 100$, the output of the Dense layer is 512, then the size of weights of this Dense layer is $10,000 \times 512$. We stack two Dense layers with one output layer of 157 classes. The network structure is listed in Table 1.

Table 1: The Structure of Fully Connected Neural Network

Layers	Input Size	Output Size	Kernel (Weight) Size
Dense	10,000	512	$10,000 \times 512$
Dense	512	128	512×128
Output	128	157	128×157

Hierarchical Fully Connected Neural Network. For hierarchical classification, the input remains the same. Both the meta-class model and the leaf-class model have the same network structure as listed in Table 1 except that the output layer of the meta-class is 5 instead of 157. Accordingly, the weight size is 128×5 of the meta-class model.

Residual Convolutional Neural Network. Convolutional Neural Network (CNN) models have been demonstrated performing well in the NLP tasks of sentence classification [8] [26]. A CNN model consists of layers of convolutions with non-linear activation

function such as *ReLU* or *tanh*. In a CNN model, convolutions over the input layer are used to compute the output. As illustrated by Zhang [27], each region of the input is connected to a neuron in the output. Then each layer applies a filter to detect higher-level features. These features are further through a pooling layer to form a univariate feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence.

To minimize the gradient vanishing problem when a CNN model grows with deep layers, He et al.[15] add residual skip connections or identity mapping to convolutional layers. The input to a layer $F(X)$ is added to convolutional output as a combined input to the next layer, $y = F(X, \{W_i\}) + W_s X$. This structure allows the gradient to be propagated without loss of representations. It is considered the skip connection and the convolutional layer form a residual block layer.

In this paper, we apply the *Full pre-activation* structure (shown in Figure 11) proposed by He et al. [16] as our Residual Block structure. The Residual Block contains two convolutional layers. A *Batch Normalization-Activation* layer is placed in-between two convolutional layers. As discussed in the paper [16], 1×1 convolution can be useful when there are fewer layers, thus we choose the 1×1 convolution layer as the skip connection method. Based on the Residual Block structure, we build our 20-layer residual convolutional neural network model shown in Table 2.

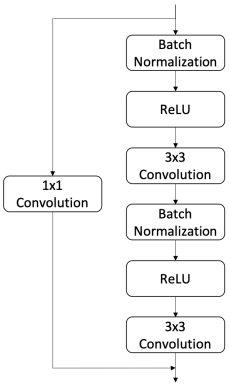

Figure 11: Structure of the Residual Block

6 THE EVALUATION

The evaluation focuses on the assessment of the classification performance. We compare the metrics of training five models with different sizes of training data and testing the models with data collected at different time spans.

6.1 The Data Setup

The dataset contains 659,421 samples. We split the data with a ratio of 80%:20% for the training set and the test set. We further partition the training set into shards of 40,000 samples for each shard. Likewise, we partition the test set into shards with 4,000 samples in each shard. Hence, the usage of the dataset is in the unit of a shard. We setup experiments of running 5 models on two data settings: (1) one shard of the training set with one shard of test set; and (2) two shards of the training set with one shard of the test set.

In addition to the 659,421 samples, we also test the best performing model using the data collected in a different time period that contains 35,663 valid samples.

Table 2: Structure of Residual CNN

Layers	Kernel	Output Size
Convolution	$\begin{bmatrix} 3\times3, 32 \end{bmatrix} \times 1$	100×100
Residual Block	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix} \times 1$	100×100
Residual Block	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	50×50
Residual Block	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	25×25
Residual Block	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	13×13
Residual Block	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	7×7
Dense		157

6.2 The Model Assessment

The evaluation defines model assessment metrics as Precision and Recall. For a multi-classes classification model, the Precision and Recall score should be calculated for each class and take the average score as the final score. There are two averaging methods, *micro* and *macro*. The macro method considers every class has the same weight, while in the micro method every data has the same weight. The calculation is as shown below, P_i is the Precision of the ith class, and R_i is the Recall of the ith class. l is the total number of classes.

$$
\begin{aligned}
P_{macro} &= \frac{1}{l}\sum_{i=1}^{l} P_i \quad P_i = \frac{TP_i}{TP_i + FP_i} \\
R_{macro} &= \frac{1}{l}\sum_{i=1}^{l} R_i \quad R_i = \frac{TP_i}{TP_i + FN_i} \\
P_{micro} &= \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} TP_i + \sum_{i=1}^{l} FP_i} \\
R_{micro} &= \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} TP_i + \sum_{i=1}^{l} FN_i}
\end{aligned}
\tag{6}
$$

where

$TruePositive(TP)$: the real label is positive and the predicted label is also positive;

$FalsePositive(FP)$: the real label is negative and the predicted label is positive;

$TrueNegative(TN)$: the real label is negative and the predicted label is also negative;

FalseNegative(*FN*): the real label is positive and the predicted label is negative.

The above metrics focus on if the classification is correct or not. Log Loss measures the distance between the predicted label and the real label. It takes the prediction probability for each class of a model as the input and outputs a log loss value as calculated in Eq 7. The lower the log loss value close to zero, the better performance of a model is. l is the total class number, y_i is the real label and p_i is the predicted probability of class i.

$$LogLoss = \sum_{i=1}^{l} y_i log(p_i) + (1 - y_i) log(1 - p_i) \quad (7)$$

6.3 The Experiment Results

The first set of experiments evaluate the classification performance of the five models. The training data sets are of one shard (40,000 samples) and two shards (80,000 samples) respectively. The test data set is one shard of 4,000 samples. Both the training set and test set are from data samples collected within the same span of time. Table 3 list the metrics measured for 5 classification models.

Table 3: Classification Performance on Five Models

Models	Training Subset	Metrics			
		Precision		Recall	
		Micro	Macro	Micro	Macro
Hierarchical FCNN	40,000	0.633	0.247	0.633	0.214
	80,000	0.686	0.332	0.686	0.288
FCNN	40,000	0.662	0.276	0.662	0.236
	80,000	0.689	0.281	0.689	0.233
Hierarchical Naive Bayes	40,000	0.746	0.439	0.746	0.367
	80,000	0.719	0.375	0.719	0.288
Naive Bayes	40,000	0.734	0.358	0.734	0.254
	80,000	0.700	0.296	0.700	0.194
Residual CNN	40,000	0.754	0.420	0.754	0.389
	80,000	**0.787**	**0.510**	**0.787**	**0.444**

Training sample size. By doubling the training sample size, we observe three models improve the classification performance, including FCNN, Hierarchical FCNN and Residual CNN.

Both Naive Bayes and Hierarchical Naive Bayes metrics decrease. As presented in section 5.1, we obtain 19,607 dimensions of input features from the request description of 40,000 training set and 32,698 dimensions of word token one-hot encoding from 80,000 training set. The observation from this experiment indicates increasing the feature size impacts the performance of our implementation of Naive Bayes classifier. Our Naive Bayes classifier learns one shard of the training set more linearly separable than the doubled size of the training set. In comparison, the word embedding method applied allows the FCNN and the Residual CNN classifiers remain the same feature size of 100 dimensions of word token vectors regardless of the size of training data.

Hierarchical vs Non-hierarchical. Hierarchical classification improves the marginal performance of Naive Bayes classifier. For FCNN and Residual CNN, they both perform better than hierarchical classification. In all experiments, Residual CNN outperforms other models. Hierarchical classification overall has lower performance than non-hierarchical classification. The benefit of introducing meta-class through the hierarchical classification method is observing the source of classification errors. Figure 12 shows the classification results with 5 meta-classes. It indicates the classification errors are mainly from the fact that class 2 and 4 are misclassified to class 1; class 1, 3, and 4 are misclassified to class 2. We also observe from the experiments that all the five classifiers produce over 80% precision for the 5 meta-class classification. The precision is higher than the classification performance of 157 classes. Due to the space limitation, we skip the values in details.

Table 4: Classification Performance of Blind Test

Models	Metrics			
	Precision		Recall	
	Micro	Macro	Micro	Macro
Hierarchical Fully connected NN	0.650	0.244	0.650	0.192
Fully connected NN	0.689	0.259	0.689	0.214
Hierarchical Naive Bayesian	0.678	0.251	0.678	0.201
Naive Bayesian	0.726	0.295	0.726	0.256
Residual Network	**0.764**	**0.417**	**0.764**	**0.352**

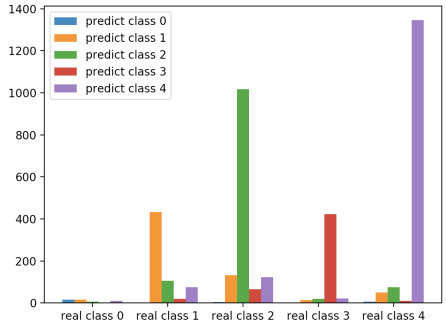

Figure 12: Distribution of predicted classes vs real classes

Blind Test. The second set of experiments run a blind test. We select the best performing model trained from each of the 5 classifiers and further test them using the whole 35,663 data samples collected from a different span of time than the first set of experiments. The result is shown in Table 4. Two classifiers, Naive Bayes and Residual CNN produce better classification performance than other three classifiers. Still, Residual CNN performs the best on the blind test data.

Log loss. We further evaluate the Residual CNN performance on the blind test data using log loss to measure the distance between the predicted label and the real label. The result is listed in Table 5. When applied to different test data, the Residual CNN has marginal log loss change.

Inference time. We further measure the inference time taken on the 4,000 test data set. Note that the classifiers of Naive Bayes and Hierarchical Naive Bayes run on a CPU node while other three neural network models run on a GPU node. Therefore the comparison between Naive Bayes models and neural network models should not be evaluated against the absolute values. Instead, we

Table 5: Log Loss of Residual CNN

Models	Test Data Size	Log Loss
Residual CNN	4,000	1.152
	35,663	1.192

observe the hierarchical classification introduce approximately 10 times inference computing delays. The inference time taken by Residual CNN is over 20 times than FCNN.

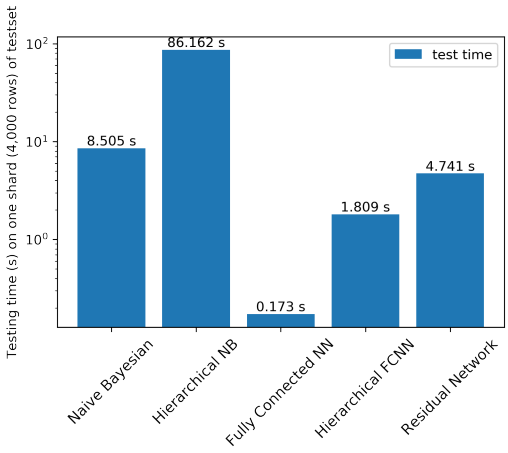

Figure 13: Inference time on the test data of 4,000 samples

7 CONCLUSION

In this paper, we present a machine learning based method of natural language classification task for a real-world application. We carry out a rigorous analysis of the dataset and design a feature engineering process that select and extract features with statistical evidence. We apply two word-encoding techniques and develop 5 classification models. This hybrid machine learning method produces benefits, namely (1) generating suitable labels for supervised learning; (2) clustering data samples into meta-class for training and initializing models to improve classification performance over unbalanced data samples;(3) producing the best performing model through comprehensive experiments and evaluation; and (4) understanding the source of error with the hierarchical classification method. The experiments setup different sizes of training data and test data. The observation shows the residual convolutional neural network model produces the best performance over other classifiers. We also observe that Naive Bayes model with the one-hot encoding of word tokens performs reasonably well with the limitation of handling increasing feature sizes. A simple two-layer fully connected neural network model has the advantage of fast inference time. It remains our future work to explore newly published word embedding model to study the effects of word embedding on classification performance.

REFERENCES

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: ordering points to identify the clustering structure. In *ACM Sigmod record*, Vol. 28. ACM, 49–60.

[2] Jeff A Bilmes et al. 1998. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute* 4, 510 (1998), 126.

[3] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 92–100.

[4] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[6] Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. LTP: A Chinese Language Technology Platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations (COLING '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 13–16. http://dl.acm.org/citation.cfm?id=1944284.1944288

[7] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781* (2016).

[8] G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2, 4 (01 Dec 1989), 303–314. https://doi.org/10.1007/BF02551274

[9] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. 1999. AdaCost: misclassification cost-sensitive boosting. In *Icml*, Vol. 99. 97–105.

[10] Mario A. T. Figueiredo and Anil K. Jain. 2002. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 3 (2002), 381–396.

[11] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[12] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer, 878–887.

[13] Pei-Yi Hao, Jung-Hsien Chiang, and Yi-Kun Tu. 2007. Hierarchically SVM classification based on support vector clustering method and its application to document categorization. *Expert Systems with applications* 33, 3 (2007), 627–635.

[14] Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

[15] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 630–645.

[17] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (2002), 881–892.

[18] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[20] Frederic Morin and Y Bengio. 2005. Hierarchical probabilistic neural network language model. *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (01 2005).

[21] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine learning* 39, 2-3 (2000), 103–134.

[22] Rudy Prabowo and Mike Thelwall. 2009. Sentiment analysis: A combined approach. *Journal of Informetrics* 3, 2 (2009), 143–157.

[23] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*. ACM, 759–766.

[24] Carlos N. Silla and Alex A. Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22, 1 (01 Jan 2011), 31–72. https://doi.org/10.1007/s10618-010-0175-9

[25] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

[26] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. MIT Press, Cambridge, MA, USA, 649–657. http://dl.acm.org/citation.cfm?id=2969239.2969312

[27] Ye Zhang and Byron C. Wallace. 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *CoRR* abs/1510.03820 (2015). arXiv:1510.03820 http://arxiv.org/abs/1510.03820