BLOG

Squadra
Machine Learning Company

Convolutional Neural Network Text Classification with Risk Assessment

"A complete overview of the whole classification system"

Natural Language Processing (NLP) has always been at the heart of Artificial Intelligence (AI) research. The ultimate objective of NLP is to read, decipher, understand and make sense of human languages in a valuable manner. Text classification is one of the most mature fields within NLP. The purpose of text classification is to automate the process of structuring textual data into one or more (predefined) categories. Text classifiers have proven to be an excellent alternative to structure textual data in a fast, cost-effective way. To take on the text classification task, we can make use of classical methods (e.g. Logistic Regression, Support Vector Machines) but also of more complex deep learning methods like the Convolutional Neural Network (CNN) or Recurrent Neural Network.
The real strength of deep learning approaches applied to the text classification problem has been uncovered not that long ago. Kim (2014) was the first to show the full potential of CNNs within the text classification framework. Since then, the CNN quickly became a hot, competitive method in the field of text classification (Sun and Gu, 2017).

A major disadvantage of text classification, however, is that one can never be sure whether a class prediction of the text classifier is correct. Because of this, human labor is still required. Although the texts do not have to be categorized manually anymore, the predicted classes following from the text classifier still have to be manually checked and adapted to the correct class if necessary. It would be very convenient if this manual effort is reduced, so that not every class prediction of the text classifier has to be checked. This manual effort can be reduced by extending the classification model in such a way that on top of each class prediction, the model also provides an indication of how sure it is about each class prediction. To achieve this, the results from the text classifier could serve as a basis for risk level prediction (RLP). The idea of RLP is that it provides a certain risk level for each class prediction, indicating whether the class prediction is risky (high risk level) or not (low risk level). This approach increases the value of the whole classification system, as the text classifier becomes aware of its own possible mistakes. This reduces the amount of manual work because the manual work can be deployed in a smarter way. The class predictions that get assigned a low risk level do not have to be manually checked, since the text classifier is convinced about the class prediction being correct. Only the class predictions that get assigned a high risk level still require human validation. Figure 1 displays the proposed classification process.

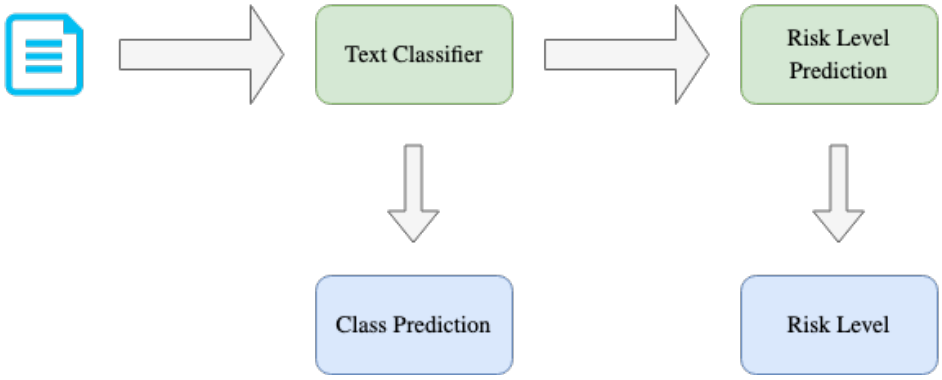

Figure 1: Classification process

Text Classification

Text classification is becoming an increasingly important part within businesses as it allows to easily get insights from textual data and enables to automate certain business processes. Automating the classification based on textual data has many applications to simplify tasks which had to be done manually before. An example application of text classifiers is sentiment analysis, where user reviews are categorized in either a positive, negative or neutral sentiment class. Another example application is product classification, where on the basis of their textual descriptions, products automatically get classified within their corresponding product class.

The field of text classification is huge, and many classification methods can be used to tackle the text classification problem (Kowsari et al., 2019). In this blog post, we will look at three different text classification approaches.

TF-IDF Logistic Regression

A relatively simple text classification method which has proven over the years that it is able to provide very competitive results is the TF-IDF logistic regression (Wang et al., 2017). This method creates TF-IDF feature vectors out of the texts, and uses these feature vectors to classify the texts within the categories by making use of the logistic regression. TF-IDF stands for Term Frequency – Inverse Document Frequency. The purpose of the TF-IDF feature vectors is to reflect the importance of a word to a particular text in the whole collection of texts. These feature vectors are calculated by taking a weighted score based on each word's occurrence in the particular text and the occurrence of this word in the whole collection of texts. The advantage of such TF-IDF scores is that they have the effect of highlighting words that contain useful information in a given text. A disadvantage is that the word order, and thus the context of words, is discarded. In turn, the text classifier is not able to recognize the contextual meaning of words or similarities between words.

The logistic regression text classifier is trained on these TF-IDF feature vectors, and can then be used to predict the classes of new texts on the basis of their TF-IDF feature vectors.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) are able to take the context and similarity of words within texts into account, in contrast to the TF-IDF logistic regression. The CNN is able to do so since it makes use of a special representation of words. This special representation of the words is called word embeddings.

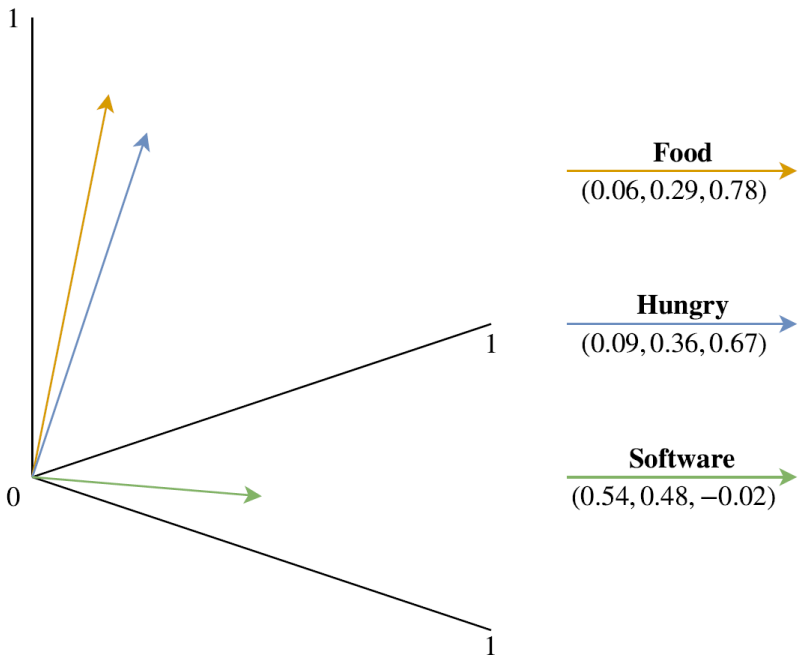

Figure 2: 3-dimensional word embeddings in the vector space

Word embeddings attempt to transform human language meaningfully into a numerical form. Word embeddings are vectors, and each vector represents the features of a certain word. The closer two word embedding vectors are to each other in the vector space, the more semantically related these words are to each other. A word embedding vector is learnt for every word in all of the texts included in the data set. In Figure 2, a simple example visualisation of 3-dimensional word embeddings of the words “food”, “hungry” and “software” is shown. Notice that the word embeddings of “food” and “hungry” are close to each other in the vector space, meaning that these words are related. On the other hand, the word embedding of “software” is far away from the other two word embeddings. Usually, word embeddings are of a much higher dimension (common dimensions: 50, 100, 200, 300), to be able to give the words more identity.

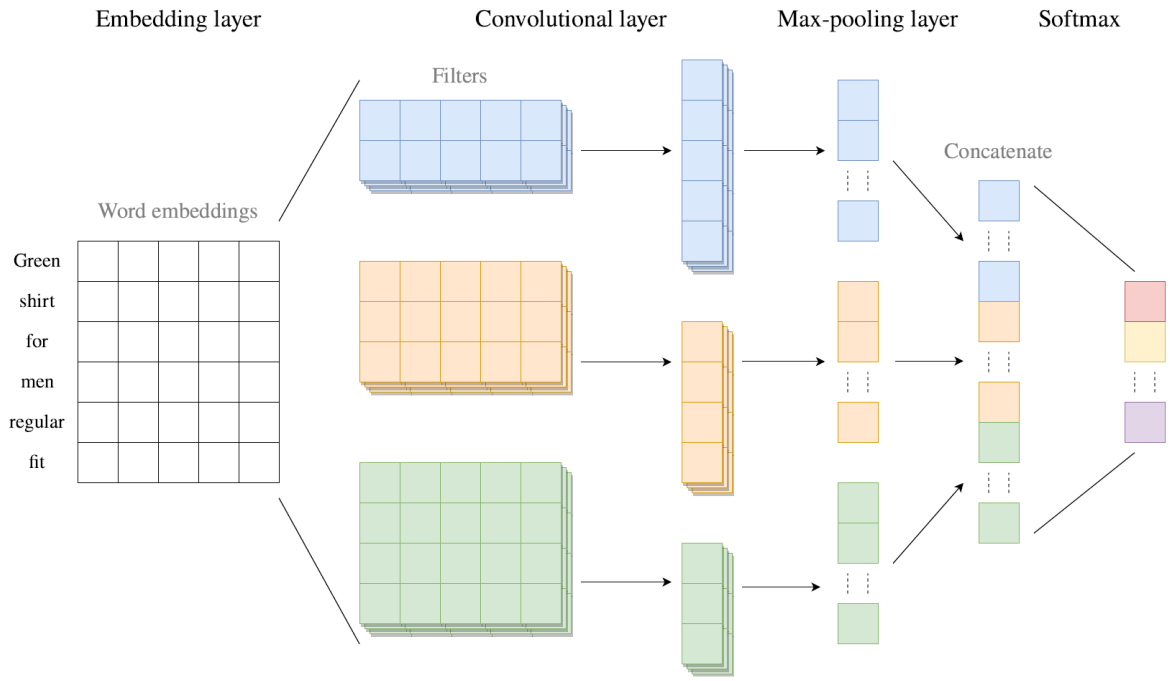

Figure 3: General CNN architecture for text classification

Now, we are going to take a look at a specific CNN architecture, depicted in Figure 3. The CNN consists of different layers, which are connected to each other. The starting layer is the embedding layer, which takes integer encoded texts as input, and initializes random or pre-trained word embedding weights. The weights are learnt during training of the CNN by updating the weights in each epoch (an epoch is a full run through the network). In Figure 3, the word embeddings are shown for the words in the text "Green shirt for men regular fit", where for illustration purposes the word embeddings are of dimension 5.

The next layer is the convolutional layer. The convolutional layer takes the texts transformed to sequences of word embeddings as input, and creates feature vectors by analysing the word embeddings for each text. This is done by the use of convolution filters. A convolution filter is a matrix that is filled with weights, that analyses multiple consecutive words in a text simultaneously, and continues like this by going through the whole text to create a feature map. This operation is performed for every text, by using multiple convolution filters, to detect various different relationships between the words. The various convolution filters may also differ from each other in height, which indicates how many consecutive words a filter considers simultaneously in each step. To obtain the feature vectors, the feature maps following from the convolution operation all get added a bias term, and an activation function is applied to add non-linearity (ReLU is the most commonly used activation function). This non-linearity enables to learn more complex relationships.

The pooling layer takes the variable-length feature vectors of the convolutional layer as input, and creates fixed-length vectors out of them. By creating these fixed-length feature vectors, the less-relevant local information is removed.

The final layer of the CNN is the softmax layer. This layer takes the fixed-length feature vectors as input, where these vectors first enter a fully-connected layer. This fully-connected layer is an efficient way of learning non-linear combinations of the features. The output of this fully-connected layer are numerical values for each class. To assign a straightforward interpretation to these numbers, the softmax function is applied. The softmax function forces the output of the CNN to represent predicted probabilities for each of the classes. The class achieving the highest predicted probability is the resulting predicted class following from the CNN.

During training of the CNN, in each epoch, the weights in the embedding, convolutional and softmax layers are updated by making use of the categorical cross-entropy loss function. This process of updating the weights is called back-propagation, and is the essence of neural net training. Back-propagation is a way of propagating the total loss obtained from the forward propagation in this epoch (i.e. , this run through the network) back into the CNN to know what portion of the loss every node is responsible for, and subsequently updating the weights in such a way that minimizes the loss by giving the nodes with higher error rates lower weights and vice versa.

Soft Vote

Until now, we have seen two different text classification methods, the logistic regression and CNN text classifiers. We can try to improve on both these individual models by combining the two, creating an ensemble text classifier. These two classifiers can be combined through a so called Soft Voting (SV) classifier, where the classes are predicted by taking a weighted average of the class probabilities following from the LR and CNN text classifiers. Such a voting classifier can be useful for equally performing classifiers in order to balance out their individual weaknesses. For the SV text classifier, we have to determine the weightings to assign to the predicted probabilities of the LR and CNN classifiers. We make use of a validation set of texts, which were not used for the training of the LR and CNN text classifiers. The SV weights are determined by finding the weights that maximize the number of correct class predictions on this validation set.

When the weights have been determined, the soft vote model is ready to calculate the weighted predicted probabilities for each class. Then, the predicted class by the SV text classifier is the class achieving the highest weighted predicted probability.

Risk Level Prediction

As mentioned earlier, there is an uncertainty regarding the correctness of the class predictions following from a text classifier. For this reason, some methodology is explained on how to assess the risk of the class predictions following from a text classifier. To perform the risk assessment of the class predictions, we make use of Risk Level Prediction (RLP). This method provides a risk level for every class prediction of the text classifier. The goal of the risk levels is to provide a critical view on the class predictions of the text classifier, to indicate how 'insecure' the classifier is about that particular class prediction. In this way, only the class predictions with a higher risk level have to be manually checked, since these are the uncertain class predictions.

Methodology

In multi-class classification, when predicting the class corresponding to a text, the class achieving the highest predicted probability, which follows from the text classifier, is assigned to the text. You might say that this predicted probability could serve as an appropriate indication of how risky the class prediction is. The higher the predicted probability, the more certain the text classifier is about its prediction being correct. Although the predicted probability of the assigned class sounds as an appropriate indicator of risk on its own, it might be even more beneficial to involve more indicators of risk to predict the risk level assigned to the class prediction even better.

The idea of the RLP is to train a second classifier which takes certain risk indicators as input, and outputs either that the class prediction is `correct' or `wrong'. Based on these predictions, we assign the risk levels. Since this classifier can assign either `correct' or `wrong' to the class prediction, this is a binary classifier. To train such a second classifier, we need to split the training data set into two subsets. The first subset is used to train the text classifier, and the second subset is used to train the classifier of the RLP. The training procedure will then be as follows:

1. The text classifier is trained on the first subset of the training data in the usual way.
2. Since at this point the text classifier has been trained, we are able to apply it to predict the classes for the texts in the second subset of the training data. Now, we both have the predicted class following from the text classifier, and the true pre-assigned class. By comparing these two classes, it can be checked whether the prediction of the text classifier was correct or wrong. This binary variable, which is either `correct' or `wrong', is the target variable within the classifier of the RLP. By using risk indicators as features for this binary classifier, we train this classifier of the RLP such that it is able to predict for new texts whether the assigned class by the text classifier is correct or wrong.

We desire that the classifier of the RLP is able to accurately predict whether a class prediction of the text classifier is correct or wrong. When this is the case, we are able to accurately assign risk levels based on the predictions of this binary classifier. To achieve this goal, we have to select useful features which are able to indicate risk of a class prediction. Useful features are the top-k predicted probabilities (i.e., the k highest predicted probabilities), sentence statistics (e.g., the length of the text), features on the behaviour of the classes (e.g., F1-score of the class), etc.

RLP variants

The RLP applies a binary classifier, categorizing the class predictions by the text classifier either as `correct' or `wrong'. The classification decision of the binary classifier is based on the traditional decision boundary. When the predicted probability of the RLP is above 0.50, the prediction will be labelled as `correct', and otherwise it will be labelled as `wrong'. In this way, we can assign the risk levels to every class prediction of the text classifier. When the predicted probability of the binary classifier of the RLP is above 0.50, we assign a risk level of 1 (not risky, a reliable prediction) and otherwise we assign a risk level of 2 (risky, an unreliable prediction). That is, for the class prediction of text i, the risk level (RL) is determined as:

$$RLi = \begin{cases} 1, & if\ P(Yi = `Correct' | Xi) \geq 0.50; \\ 2, & if\ P(Yi = `Correct' | Xi) < 0.50, \end{cases}$$

where Y is the target variable of the binary classifier of the RLP and X are the features.

The mentioned RLP variant applies only one binary classifier to predict the risk levels. Instead of using only one binary classifier, we can also make use of two binary classifiers and extend the idea of the previous variant further. A simple binary form when combining two binary classifiers is when a class prediction gets assigned a risk level of 1 when both binary classifiers predict the class prediction to be correct (based on the decision boundary of 0.50), and a risk level of 2 is assigned otherwise.

We can extend this idea to RLP variants which are able to assign more than two different risk levels. When the two binary classifiers both categorize the class prediction by the text classifier as `correct' with much certainty, a low risk level would fit well. When one binary classifier assigns `correct' with much certainty and the other with less certainty, it is still very likely that the predicted class by the text classifier is correct, but it is a bit more risky. We can continue the reasoning in this way. An example variant of the RLP, assigning seven different risk levels when combining the results of two different binary classifiers, is visualized in Figure 4. Here, for text i, pi = P(Yi = `Correct'|Xi) is the predicted probability of the first binary classifier and qi = P(Zi = `Correct'|Xi) is the predicted probability of the second classifier.

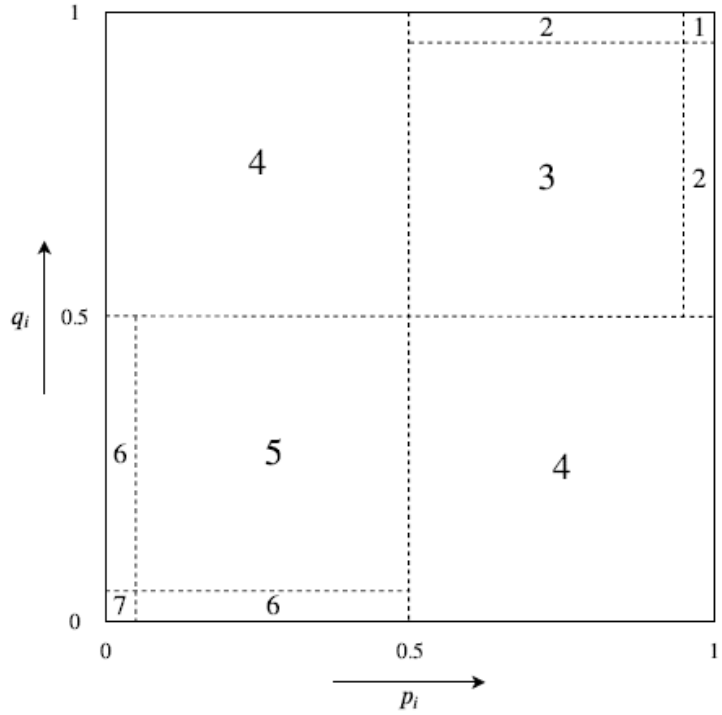

Figure 4: Visualization of the risk level assignment for the seven RL variant

The different variants of the RLP discussed here are examples of how we can assign the risk levels. Of course, you could think of many more variations on how to assign the risk levels.

Classification System

A complete overview of the whole classification system as explained in this blog post is shown in Figure 5. In the figure, the CNN text classifier is used for the class predictions. As explained, first, the CNN text classifier is trained on the first split of the training data. Then, on the second split of the training data, the binary classifier of the RLP is trained. Finally, we can apply both the text classifier and RLP on new texts to predict the class and provide a risk level alongside this class prediction.

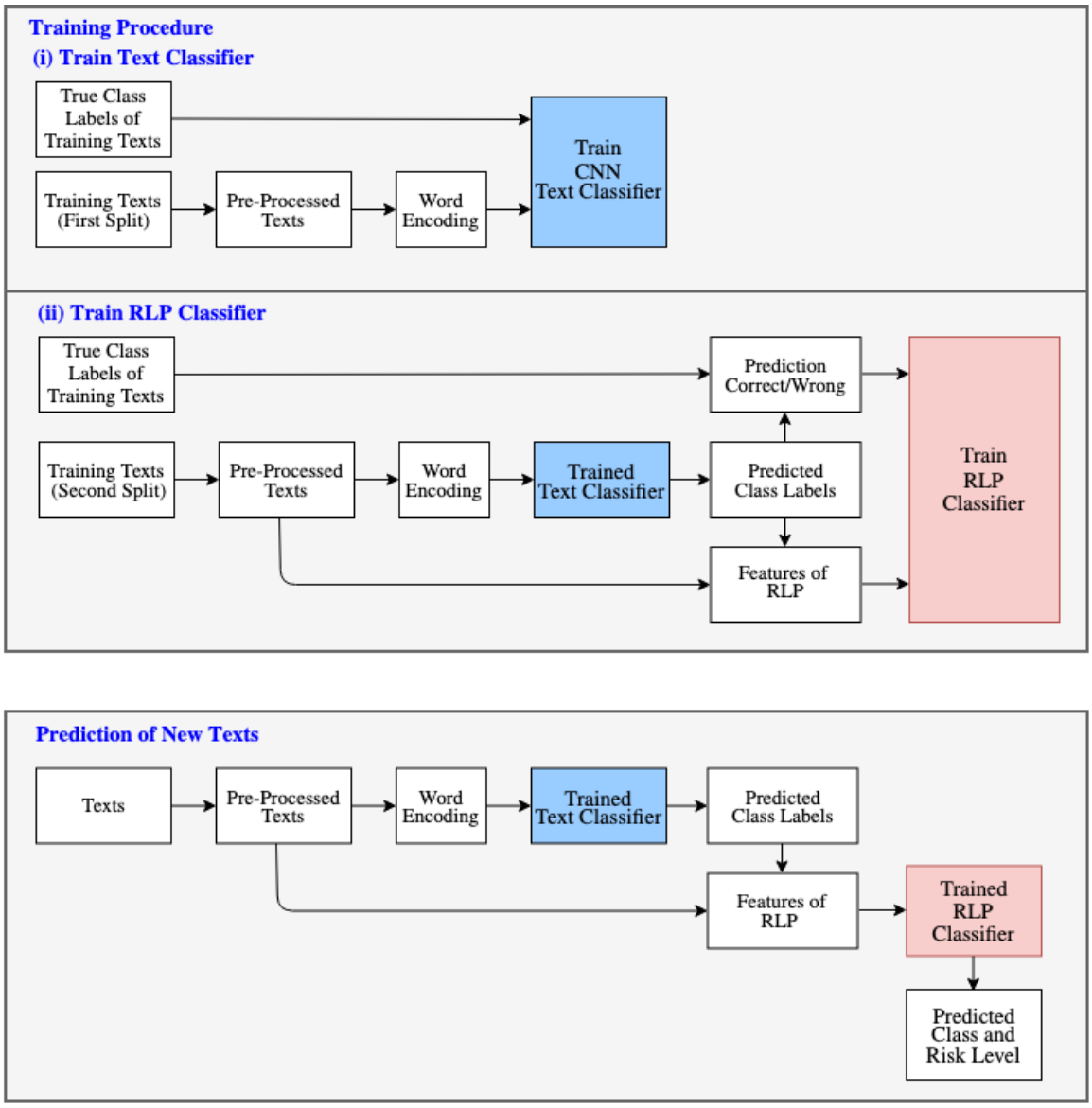

Figure 5: Classification system (using CNN text classifier)

References

Kim, Y. (2014). *Convolutional Neural Networks for Sentence Classification*. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Kowsari, K., Meimandi, K.J., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D. (2019). *Text Classification Algorithms: A survey*. Information, 10 (4), 150.

Sun, S. and Gu, X. (2017). *Word Embedding Dropout and Variable-Length Convolution Window in Convolutional Neural Network for Sentiment Classification*. ICANN 2017 (2), 40-48.

Wang, Y., Zhou, Z., Jin, S., Liu, D. And Lu, M. (2017). *Comparisons and Selections of Features and Classifiers for Short Text Classification. IOP Conference Series: Materials Science And Engineering, 261.*

Wang, Zhou, Jin, Liu, Lu (2017) Comparisons and selections of features and classifiers for short text classification.