Interpreting deep text quantification models

by

Yun Qi Bang

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Wallapak Tavanapong, Co-major Professor
Zhu Zhang, Co-major Professor
Qi Li

The student author and the program of study committee are solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2022

DEDICATION

I would like to dedicate this thesis to my mother, Rebecca Chin, for her ever-lasting love and support.

TABLE OF CONTENTS

Page

LIST OF TABLES iv

LIST OF FIGURES v

ACKNOWLEDGMENTS vi

ABSTRACT vii

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. RELATED WORK 3
2.1 Quantification Methods 3
2.1.1 Classification-Based Quantification Methods 3
2.1.2 Direct Quantification 5
2.2 Interpretation Methods 6

CHAPTER 3. BACKGROUND 8
3.1 QuaNet 8
3.2 Layer-wise Relevance Propagation (LRP) 10

CHAPTER 4. INTERPRETING QUANET 11

CHAPTER 5. EXPERIMENTAL DESIGN AND RESULTS 15
5.1 Datasets 15
5.2 Experimental Settings 16
5.3 Performance Metrics 17
5.4 QuaNet's Quantification Performance 18
5.5 RQ1: What input features are important to the final class distribution prediction of QuaNet? 18
5.6 RQ2: Does sorting L by $\Pr(c|\mathbf{x})$ increase the performance of QuaNet? 22

CHAPTER 6. CONCLUSION AND FUTURE WORK 25

BIBLIOGRAPHY 26

LIST OF TABLES

Page

Table 4.1 Notation used in Chapter 4 14

Table 5.1 Statistics of the five datasets used for experiments. 15

Table 5.2 Performance of QuaNet and its underlying classifier on different datasets . . 18

Table 5.3 Performance of QuaNet VS Small-NN 22

Table 5.4 Performance of different QuaNet models on binary quantification tasks . . . 23

Table 5.5 Performance of different QuaNet models on multi-class quantification tasks . 24

LIST OF FIGURES

Page

Figure 3.1 The architecture of QuaNet 9

Figure 4.1 Visual example of Algorithm 1 on a two-class quantification. 13

Figure 5.1 Distribution of documents of PAPT datasets 16

Figure 5.2 Percentage of contributions of each input component of QuaNet on different test datasets 19

Figure 5.3 Percentage of contributions of each input component of QuaNet on samples with different distributions 20

ACKNOWLEDGMENTS

I would like to take this opportunity to express my heartfelt thanks to those who helped and supported me throughout this journey.

First and foremost, I would like to thank Dr. Wallapak Tavanapong for her guidance, patience, and support throughout this research and the writing of the thesis. Her insights and words of encouragement can always induce great motivation and renew my hopes for completing my graduate education.

I would also like to thank both of my major professors, Dr. Wallapak Tavanapong and Dr. Zhu Zhang, for their advice, kindness, and understanding throughout my journey in graduate school. Without whom I would not be able to grow into the researcher I am today.

I would additionally like to thank Mohammed Khaleel for all his effort and contributions to this work. His motivating encouragement and thought-inspiring discussions helped tremendously in completing this work.

ABSTRACT

Quantification learning is a relatively new deep learning task. Differing from a classic classification problem where the class of a single instance is predicted, a quantification model predicts the distribution of classes within a given set of instances. Quantification learning has applications in various domains. For example, in designing political campaign ads, it is important to know the proportion of different aspects voters care about. QuaNet is a recent deep learning quantification model that was shown to achieve good quantification performance. Like many deep learning models, there is no explanation about the contributions of different inputs QuaNet uses to predict a class distribution. In this study, we propose a method to provide such an explanation, which is important to increase users' trust in the model. Our method is the first work on interpreting deep learning quantification models.

CHAPTER 1. INTRODUCTION

Quantification is the task to estimate the prevalence of classes of interest given a set of unlabeled instances [31]. Being able to predict the class distribution or class ratios is important for numerous domains such as social sciences, epidemiology, healthcare, and market research [31]. For example, in healthcare, the proportion of patients in different age groups affected by a specific disease is important for the development of appropriate treatment and prevention. In commerce, quantification is useful in a market survey to determine customers' preferences on different aspects of a product.

Quantification learning is formally defined as "given a labeled training set, induce a quantifier that takes an unlabeled test set as input and returns its best estimate of the class distribution" [12]. Quantification learning has been solved by different methods, namely, Classify and Count (CC) [13], an ensemble-based method [33], distribution matching [38], and direct quantification [27, 26, 17]. Only recently, deep learning quantifiers were proposed [35, 10].

Deep learning models have been shown to achieve excellent performance in a variety of tasks such as image captioning [18, 42], object detection [44, 21], and time series forecasting [24]. There are plenty of deep learning models for classification tasks [22, 40, 20], but there exist only two deep learning models for quantification tasks: Recurrent Neural Network Architecture for Quantification (QuaNet) [10] and Deep Quantification Network (DQN) [35]. Both approaches showed better performance than non-deep learning models for quantification tasks. As black box models, they do not provide information about what features neurons extract and their influence on the final class distribution prediction.

Explainable AI (XAI) has been a growing field of study as artificial intelligence, particularly machine learning, is widely adopted in real-life domains [29]. The main motivations of XAI are to 1) increase end users' trust towards the models' prediction and 2) provide developers some insight

on whether the model is working as intended. To date, there are many interpretation methods designed for classification tasks [19], but none exists for quantification tasks. This motivates our study to interpret deep quantification models.

The two deep quantification models, QuaNet [10] and DQN [35], are very different in their design. Hence, it is difficult to design a unified approach that applies to both models. In this work, we focus on designing an interpretation method for QuaNet since it was the top-performing method among thirteen different quantification methods in one of the datasets, and ranked fourth in performance based on the average absolute error on eleven different datasets in a recent study [31]. Furthermore, a public Python library, `QuaPy` [30], was created by the authors of QuaNet and is readily available.

Our interpretation method is built on Layer-wise Relevance Propagation (LRP) [4] to identify the relevance of an input dataset to the final class distribution prediction. LRP is a well-known interpretation method that has been applied for the interpretation of text and image classification models [4, 2]. LRP was also used in designing an interpretation method for Long Short-Term Memory (LSTM) recurrent networks [3]. LSTM is a component of QuaNet, which makes LRP suitable for interpreting QuaNet. Additionally, recent studies [41, 1] showed that LRP outperformed several other well-established explanation methods. Our unique contributions are as follows.

- The first interpretation of a deep quantification model, QuaNet, using LRP.
- Explanation of the QuaNet models on three public datasets: IMDB [25], YELP [45], and AG-NEWS [45], and one political science dataset [34].

The remaining of this paper is organized as follows. Chapter 2 provides a brief overview of related works categorized in both quantification and interpretation methods. Chapter 3 provides background knowledge on QuaNet and LRP. Chapter 4 introduces our proposed interpretation method. In Chapter 5, we show the experiments, results, and findings. Finally, Chapter 6 concludes our study and states the intended future work.

CHAPTER 2. RELATED WORK

In this chapter, we present an overview of related quantification learning methods and deep learning model interpretation.

2.1 Quantification Methods

We group existing quantification methods into two categories: classification-based methods and direct quantification methods. Classification-based methods utilize a given classifier, as an initial step, to classify the individual samples and then post-process the classification results to reach the quantification prediction. Whereas direct quantification methods predict the final quantification in an end-to-end manner without the need to use the classifier.

2.1.1 Classification-Based Quantification Methods

The naïve solution to quantification is the Classify and Count (CC) method [13], where a classifier is trained, and the prediction result is used to count the documents predicted in each class. More formally, given a classifier f and a set of unlabeled documents D^U, the predicted prevalence of class c by this method is

$$\hat{p}_c^{CC}(D^U) = \frac{|\{\mathbf{x} \in D^U | f(\mathbf{x}) = c\}|}{|D^U|} \tag{2.1}$$

with $\sum_{c \in C} \hat{p}_c^{CC}(D^U) = 1$, where C is the set of classes, and $|D^U|$ denotes the cardinality of set D^U, that is, the number of documents in D^U.

Forman et al. proposed Adjusted Classify and Count (ACC) [13]. ACC adjusts the predicted class prevalence by the CC method with the true positive rate for class c (tpr_c) and the false positive rate for the class (fpr_c). These values are obtained from standard cross-validation on the

labeled training set, D^L. The predicted prevalence of class c with ACC is defined as

$$\hat{p}_c^{ACC}(D^U) = \frac{\hat{p}_c^{CC}(D^U) - fpr_c}{tpr_c - fpr_c} \tag{2.2}$$

Probabilistic Classify and Count (PCC) and Probabilistic Adjusted Classify and Count (PACC) [6] are as their names suggest, the probabilistic versions of CC and ACC, respectively. PCC and PACC are designed to use the posterior probabilities from the underlying classifiers as they contain richer information than the classifier's final outcomes. Let $\Pr(c|\mathbf{x})$ be the posterior probability of document $\mathbf{x}$ belonging to class c by the classifier f. The predicted prevalence of class c with PCC and PACC is defined below.

$$\hat{p}_c^{PCC}(D^U) = \frac{\sum_{\mathbf{x} \in D^U} Pr(c|\mathbf{x})}{|D^U|} \tag{2.3}$$

$$\hat{p}_c^{PACC}(D^U) = \frac{\hat{p}_c^{PCC}(D^U) - fpr}{tpr - fpr} \tag{2.4}$$

Saerans et al. [38] used the Expectation-Maximization (EM) algorithm [7] to update the posterior probabilities thus adjusting the classifier's outcome according to the distribution drift without the need to re-train the classifier. Esuli et al. [10] introduced QuaNet as a deep-learning quantification model that is built on top of a classifier. QuaNet utilizes a probabilistic classifier to obtain document representations and posterior probability along with CC, ACC, PCC, and PACC to get the final quantification prediction. We discuss QuaNet in the next chapter.

The Quantification Trees method is proposed by Milli et al [27] to optimize the classifier (Decision Tree or Random Forest) by minimizing the L2 norm of the difference between the predicted and the true class prevalence of all the classes. Esuli and Sebastiani [11] proposed a learning algorithm for quantification by adopting Support Vector Machine (SVM) for Multivariate Performance Measures (SVMperf). Q-measure (analogous to F-measure in classification) is proposed by Barranquero et al [5] as a loss function for quantification. An ensemble quantifier is proposed by Pérez-Gállego et al [33] where k base quantifiers are trained with different randomly drawn samples, and at testing, all the quantifiers are applied on the test dataset and a final prediction is aggregated. HDy is a probabilistic binary quantification method proposed by

González et al [14] by minimizing the Hellinger Distance between two classifier output distributions, one from test data and the other from validation data. HDy returns the mixture parameter α that best fits the validation distribution to the test data distribution as the predicted prevalence of the positive class.

2.1.2 Direct Quantification

Although a perfect classifier yields a perfect quantification result, an imperfect, good classifier does not necessarily give a good quantification result. Consider a binary classification with ten instances where five instances are of the positive class and the other five are of the negative class. The true class ratio is 1:1. Suppose classifier A's prediction gives only two false positives in the positive class, resulting in an accuracy of 80%. Suppose classifier B's prediction gives two false positives for each class, resulting in its accuracy of 60%. Even though classifier A has higher classification accuracy, using it for predicting the class ratio results in the class ratio of 7:3, which is worse than the quantification result of 1:1 by classifier B. Furthermore, having a perfect classifier to achieve good quantification results is not possible for most classification tasks. This motivates the adaptation of quantification methods that predict the quantification without the need to train a classifier.

Similar to HDy, HDx is another quantification method from [14] where the data itself is used to calculate the distance, instead of the classifier's posterior probabilities. Thus HDx does not utilize a classification model. ReadMe is another direct quantification method developed by Hopkins and King [15] to solve text quantification problems. ReadMe performs the following steps for a user-defined number of iterations on the training dataset. First, it randomly selects k words to form 2^k word patterns and calculates the proportion of these word patterns across different classes. Then, ReadMe uses the word patterns and the proportion to estimate the class prevalence of the test dataset. ReadMe2 [17] is an improved version of ReadMe. Instead of randomly choosing words as features, it uses pre-trained word embedding to produce the document-feature matrix. ReadMe2 then applies an optimization-based dimension reduction to

obtain a lower-dimension feature matrix. The feature matrix is used in place of the word patterns and the proportion to estimate the class prevalence.

DQN [35] is the first deep learning-based quantification model that is trained in an end-to-end manner. DQN is trained by splitting the training dataset into subsets. Then for each subset, DQN learns to obtain the feature vector of each document. DQN then merges all the feature vectors into one feature vector and passes it through the final fully connected layer to produce a prediction for each subset. The class distribution prediction of the whole training dataset is obtained by averaging the outcome of all subsets. DQN is trained with stochastic gradient descent to optimize the Jensen-Shannon Divergence [9]. During testing, DQN divides the test dataset into subsets to obtain feature vectors for each subset. All of these subsets' feature vectors are then used to produce a class prevalence prediction. The final prediction of the test dataset is obtained by averaging the predictions from all the subsets.

2.2 Interpretation Methods

First, we clarify the terminology used for interpretation in this text. Global or model interpretation is used to explain the average behavior of a method, whereas local or prediction interpretation is used to explain individual prediction [28]. In this paper, we mainly focus on the local interpretation of the deep quantification models. In other words, our explanation focuses on explaining the final prediction of a model based on the given test dataset.

Next, we explain our decision on utilizing LRP by exploring the pros and cons of different existing interpretation methods. Local interpretation methods on deep learning models can be grouped into five main categories [19]: probe internal representation, input perturbation, signal backpropagation, gradient-based, and attention-based approaches.

Approaches that probe internal representation focus on examining a neuron's input and output signal in a layer of the neural network. For example, the method proposed by Jacovi et al [16] examines a Convolutional Neural Network (CNN) by deriving a unique identity for each of its filters based on the most relevant n-grams. Their approach uses the activation score of each

convolutional filter as the relevance score of the words, meaning that the interpretation cannot differentiate between positive or negative contributions. This method was also mainly developed for CNNs.

Input perturbation approaches find the most relevant features to the final prediction by comparing the performance changes caused by the perturbation of the model's input. LIME [37] and the Leave-one-out [23] method are examples of this category. This category is difficult to be implemented for quantification models, as the input to quantification is a set of documents, and quantification features of deep quantification models cannot be determined naturally, thus it is unclear how perturbation of input can be performed.

Signal backpropagation approaches like LRP [4] and Class Activation Map (CAM) [46] operate by distributing the class probability score of a target class via backpropagation from the output layer to the input layer of a classifier. We have chosen to utilize LRP for our study. We did not consider CAM as it requires a global pooling layer followed by a fully connected layer as the last layer before the output layer, thus a restriction on model architecture. In contrast, LRP is not limited to CNNs and can be applied to a variety of deep learning models.

Saliency Map [8] and Gradient-Class Activation Map (Grad-CAM) [39] are gradient-based approaches. These approaches are similar to signal backpropagation approaches with the exception that they use only gradient information. This can lead to the possibility of only reflecting the function locally and failing to identify the contribution of input features to the prediction [1].

Lastly, attention-based approaches [32, 43] are applicable to attention-based models, where the learned attention vector is used to find the top contributing features. We do not consider this approach as it is restricted to attention-based models, and both existing deep quantification models are not attention-based.

CHAPTER 3. BACKGROUND

3.1 QuaNet

Our focus is on interpreting QuaNet. Here, we provide an in-depth background on the architecture of QuaNet, and how its input is processed.

In Chapter 2.1, we categorized QuaNet as a classification-based quantification method as its input depends on a classifier. QuaNet is a Recurrent Neural Network (RNN) for quantification that is designed with the intuition that a recurrent network can observe patterns and learn to count. QuaNet architecture contains a recurrent network, followed by several fully connected layers with ReLU activation, and a final layer of size $|C|$ with softmax activation. Given a dataset D^U (set of unlabeled documents), a target class c, and a classifier f that returns posterior probabilities $\Pr(c|\mathbf{x})$, the input of QuaNet consists of five different components: (1) a list L of pairs $\langle \Pr(c|\mathbf{x}), \vec{\mathbf{x}} \rangle$ for all $\mathbf{x}$ in the dataset D^U, where $\vec{\mathbf{x}}$ is the document embedding for $\mathbf{x}$ and L is sorted by $\Pr(c|\mathbf{x})$, (2) $\hat{p}^{CC}(D^U)$, (3) $\hat{p}^{ACC}(D^U)$, (4) $\hat{p}^{PCC}(D^U)$, and (5) $\hat{p}^{PACC}(D^U)$. Figure 3.1 shows the architecture of QuaNet given $|C| = 2$.

Training Process: Let D^L be the training dataset. Given an already trained classifier f, the training procedure of QuaNet is as follows. Run the classifier on each document $\mathbf{x}$ in D^L to obtain the document representation and posterior probabilities. Apply CC, ACC, PCC, and PACC quantifiers on the classifier's result. The sorted list L (input 1) is then created and passed through the first component of QuaNet, the bidirectional LSTM (bi-LSTM) layer, to obtain a quantification embedding. Let h be the hidden size of an LSTM cell. Thus, the output of the bi-LSTM layer, quantification embedding, is a vector of $2h$ elements. The quantification embedding is then concatenated with the four quantifiers' predictions (inputs 2-5), and each quantifier's result is a vector of $|C|$ elements. The resulting vector of size $2h + 4|C|$ is given to QuaNet's second component, the fully connected layers with ReLU activation. The final layer

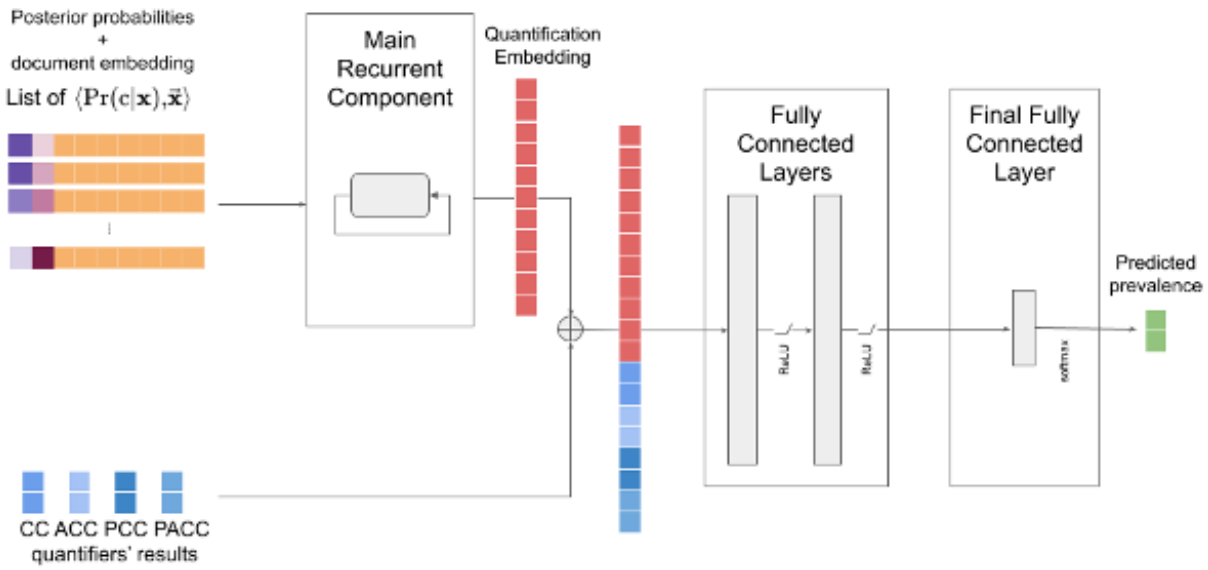


Figure 3.1 The architecture of QuaNet given $|C| = 2$; its inputs are produced by a separate classifier. QuaNet consists of three components. The main recurrent component produces quantification embedding given a list of document representations, $\vec{\mathbf{x}}$, and posterior probabilities, $Pr(c|\mathbf{x})$, from the trained classifier. We later refer to this list as L. The fully connected layers process the concatenation of quantification embedding and predictions from CC, ACC, PCC, and PACC. The final layer with $|C|$ neurons produces the final prediction.

with $|C|$ neurons and softmax activation output the final prediction. The loss function is the Mean Square Error (MSE) of the prediction compared to the ground truth. The loss function is minimized via stochastic gradient descent. QuaNet calculates the *tpr* and *fpr* values by applying the classifier f to a separate validation dataset. Finally, the *tpr* and *fpr* values are used to calculate $\hat{p}^{ACC}(D^L)$ and $\hat{p}^{PACC}(D^L)$ using Equations 2.2 and 2.4, respectively.

Testing Process: QuaNet uses the same process as its training procedure to produce its input from the testing dataset, D^U. The list L and the four quantifiers' predictions then pass through the trained QuaNet to arrive at a final prediction.

3.2 Layer-wise Relevance Propagation (LRP)

LRP was first proposed to study how a deep image classifier makes its decisions by computing the relevance flow in a neural network [4]. It uses the network weights and the neural activations created during the forward pass to propagate the output back through the network up until the input layer. Then the contribution of each pixel to the classifier's decision is visualized for users' understanding.

LRP uses the layer-wise relevance conservation principle, meaning that the relevance that is received by a neuron must be redistributed to the lower layer in equal amounts. For a deep learning model f, input x, and target class c, LRP redistributes the posterior probability of class c for x, $f_c(x)$, layer by layer, following different rules [29, 3] for different types of layers.

The benefits of LRP outweigh its time-consuming drawback for interpretation. (1) LRP is not dependent on the input type. Thus, the input can be a mix of two or more forms, for example, a list of words and numerical scores. (2) The application of LRP is straightforward and does not require the training of an external model to derive the explanation. (3) LRP is applicable to different model architectures [4, 2]. (4) Finally, the output relevance scores are signed values, which differentiate whether a highly relevant feature contributes positively or negatively to the final prediction.

CHAPTER 4. INTERPRETING QUANET

In this chapter, we explore two research questions.

- RQ1: What inputs are important to the class distribution prediction by QuaNet?
- RQ2: Does sorting L by $Pr(c|\mathbf{x})$ increase the performance of QuaNet?

We designed Algorithm 1 to answer the research questions. Due to the aforementioned benefits of LRP, Algorithm 1 uses LRP to derive a relevance score for each input of the QuaNet model to the final prediction. A large relevance score means the input is more important. A positive score means that the input contributes positively to the prediction, and a negative score means otherwise. A score of 0 means that the input does not contribute anything.

Algorithm 1 is based on the QuaNet implementation by the authors of QuaNet [10], where the main recurrent component is a bi-LSTM layer, followed by two fully connected layers with ReLU activation as the second component, and a final layer with $|C|$ neurons with softmax activation. Table 4.1 shows the notations used in this chapter. Given a class c and a classifier f that returns posterior probabilities $\Pr(c|\mathbf{x})$, QuaNet takes five different components as input: (1) a sorted list L of pairs $\langle \Pr(c|\mathbf{x}), \vec{\mathbf{x}} \rangle$ for all $\mathbf{x}$ in the test dataset D^U, where $\vec{\mathbf{x}}$ is the document embedding for $\mathbf{x}$ and L is sorted by the values of $\Pr(c|\mathbf{x})$, (2) $\hat{p}^{CC}(D^U)$, (3) $\hat{p}^{ACC}(D^U)$, (4) $\hat{p}^{PCC}(D^U)$, and (5) $\hat{p}^{PACC}(D^U)$. L is an input of the bi-LSTM layer to obtain a quantification embedding. The input to the first fully connected layer is the concatenation of the quantification embedding and input 2-5.

The required inputs to Algorithm 1 are the QuaNet model to be interpreted, and the list of five input components of QuaNet as defined previously. Figure 4.1 shows the overall interpretation process of Algorithm 1. In this algorithm, Step 1 runs the QuaNet model on the input to obtain the intermediate input and output of each layer. In Step 2, the relevance scores of

Algorithm 1 Interpreting QuaNet

Input: Trained QuaNet model and S^U
Output: Relevance score of each input component of QuaNet in S^U
Notation: See Table 4.1.

1: Run QuaNet on S^U to get intermediate input and output of each layer
2: $R_{out}^{fclayer3} \leftarrow$ List of neuron activations of $fclayer3$ $\ell \in [fclayer3, fclayer2, fclayer1]$
3: $R_{in}^{\ell} \leftarrow \texttt{LRP}(\ell, I_\ell, O_\ell, R_{out}^{\ell})$
4: $R_{out}^{\ell'} \leftarrow R_{in}^{\ell}$ $\#\ell'$ is the next ℓ; this step is excluded in the last iteration
5: $R_{out}^{bi-LSTM} \leftarrow R_{in}^{fclayer1}[0{:}2h]$ #list of the first $|O_{bi-LSTM}|$ elements of $R_{in}^{fclayer1}$
6: $R_L \leftarrow \texttt{LRP}(\text{bi-LSTM}, I_{bi-LSTM}, O_{bi-LSTM}, R_{out}^{bi-LSTM})$
7: $R_{CC}, R_{ACC}, R_{PCC}, R_{PACC} \leftarrow R_{in}^{fclayer1}[2h{:}(2h + 4|C|)]$ #list of all elements after the $|O_{bi-LSTM}|$-th elements in $R_{in}^{fclayer1}$
8: **return** $R_{CC}, R_{ACC}, R_{PCC}, R_{PACC}, R_L$

the last layer, $fclayer3$, are set as its neuron activations. Steps 3-6 iterate over the three QuaNet's fully connected layers, from the last layer, $fclayer3$, to the first fully connected layer, $fclayer1$. The relevance scores of each layer are backpropagated to the layer before it. In Step 4, `LRP` is used to calculate the relevance scores to be propagated to the previous layer, R_{in}^{ℓ}, by using the layer information of layer ℓ, the input I_ℓ, the output O_ℓ, and the relevance scores of the layer ℓ, R_{out}^{ℓ}. Step 5 then assigns the relevance scores to the propagated as the relevance scores of the layer ℓ in the next iteration. At the last iteration, the relevance scores to be propagated are for the bi-LSTM layer and the four quantifiers' results.

Recall from Chapter 3.1 that h is the hidden size of an LSTM cell, and the quantification embedding is a vector of $2h$ elements. The purpose of Step 7 and Step 9 is to allocate elements in $R_{in}^{fclayer1}$ to $R_{out}^{bi-LSTM}, R_{CC}, R_{ACC}, R_{PCC}, R_{PACC}$. Step 7 assigns the first $2h$ relevance scores as $R_{out}^{bi-LSTM}$ for Step 8. In Step 8, the relevance scores of the bi-LSTM layer are backpropagated to list L. To interpret the bi-LSTM layer of QuaNet in Step 8, we followed the implementation of [3][1]. Finally, Step 9 assigns the rest of $R_{in}^{fclayer1}$ as the relevance scores for the four quantifiers' results.

[1] https://github.com/alewarne/Layerwise-Relevance-Propagation-for-LSTMs

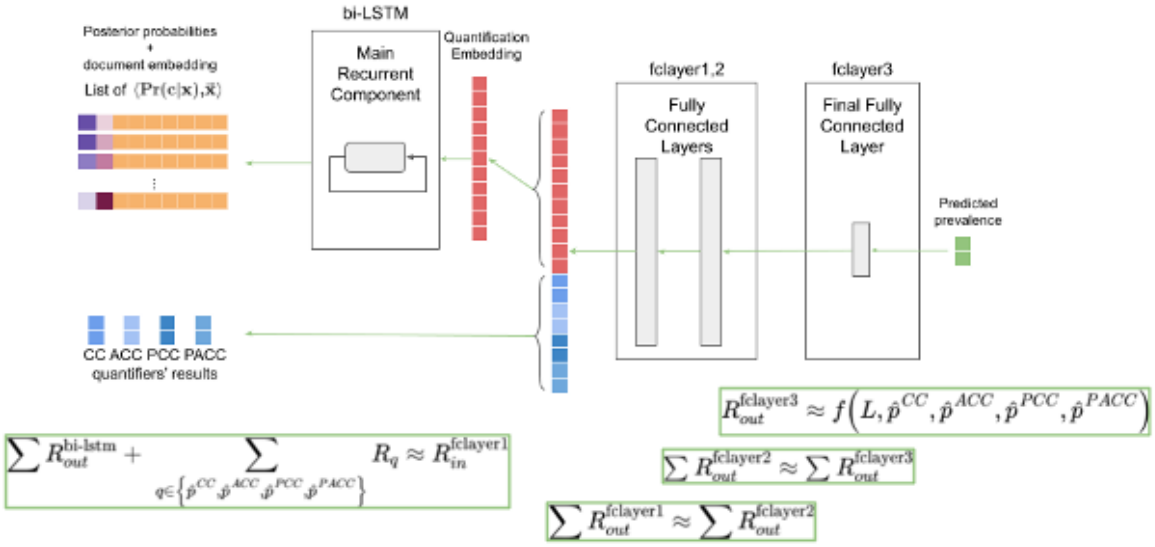


Figure 4.1 Visual example of Algorithm 1 on a two-class quantification.

Table 4.1 Notation used in Chapter 4

C	Set of classes
D	Set of documents
D^L	Set of labeled documents/training set
D^U	Set of unlabeled documents/test set
I_ℓ	Input of layer ℓ
O_ℓ	Output of layer ℓ
R^ℓ_{in}	List of relevance scores to be propagated to the layer before ℓ, $\lvert R^\ell_{in}\rvert = \lvert I_\ell\rvert$
R^ℓ_{out}	List of relevance scores for all the neurons in layer ℓ, $\lvert R^\ell_{out}\rvert = \lvert O_\ell\rvert$
$\texttt{LRP}(\ell, I_\ell, O_\ell, R^\ell_{out})$	`LRP` denotes the function that implements the `LRP` algorithm. It returns the relevance score of each element of the input of this layer l.
c	Target class
f	Trained classifier
$\Pr(c\lvert\mathbf{x})$	Class posterior probability of document $\mathbf{x}$ for class c, which is obtained from classifier f
$\vec{\mathbf{x}}$	Document representation of $\mathbf{x}$ obtained from classifier f
L	Sorted list of pairs of document representation ($\vec{\mathbf{x}}$) and $\Pr(c\lvert\mathbf{x})$ for a chosen class c for all input documents
bi-LSTM, fclayer1, fclayer2, fclayer3	QuaNet BiLSTM and its three fully connected layers
h	Hidden size of an LSTM cell
S	List of five elements: L, $\hat{p}^{CC}(D)$, $\hat{p}^{ACC}(D)$, $\hat{p}^{PCC}(D)$, $\hat{p}^{PACC}(D)$
$I_{bi-LSTM}$	Same as L
$I_{fclayer1}$	$O_{bi-LSTM} \oplus \hat{p}^{CC}(D) \oplus \hat{p}^{ACC}(D) \oplus \hat{p}^{PCC}(D) \oplus \hat{p}^{PACC}(D)$ where $\oplus$ denotes a concatenation
R_X	Relevance score of a QuaNet input component X. For example, R_{CC} is the relevance score of $\hat{p}^{CC}(D)$, where $\lvert R_{CC}\rvert = \lvert\hat{p}^{CC}(D)\rvert$.

CHAPTER 5. EXPERIMENTAL DESIGN AND RESULTS

This chapter describes the datasets, the experimental settings, and the results of the interpretation of QuaNet. The design of the experiments was guided by our two research questions.

5.1 Datasets

We used five different classification datasets as summarized in Table 5.1. Three of them are publicly available and commonly used for quantification tasks. They are also balanced datasets, i.e., having the same number of documents for each class. The Standford Large Movie Review dataset (IMDB) has movie reviews with binary sentiment classification [25]. Yelp Polarity Reviews (YELP) [45] dataset was constructed for binary sentiment classification by grouping reviews with one or two stars in a negative class and reviews with three or four stars in a positive class. AG-NEWS [45] is a balanced multi-class classification dataset. For our study, only the titles in AG-NEWS were considered, while the article and description of each document were removed. This resulted in an average of 8 words per document.

Table 5.1 Statistics of the five datasets used for experiments.

Dataset	#classes	# training documents	#testing documents	Average #words
IMDB	2	25,000	25,000	231
YELP	2	560,000	76,000	133
AG-NEWS	4	120,000	7,600	8
PAPT-Iowa	20	10,124	1,114	16
PAPT-Nebraska	20	5,132	559	17

The PAPT dataset [34], or the Policy Agenda Project Topics dataset is a highly imbalanced dataset constructed by collecting tweets from state legislators in the US. These tweets were manually classified into 20 topics by Policy Agenda Project [36]. PAPT-Iowa and PAPT-Nebraska

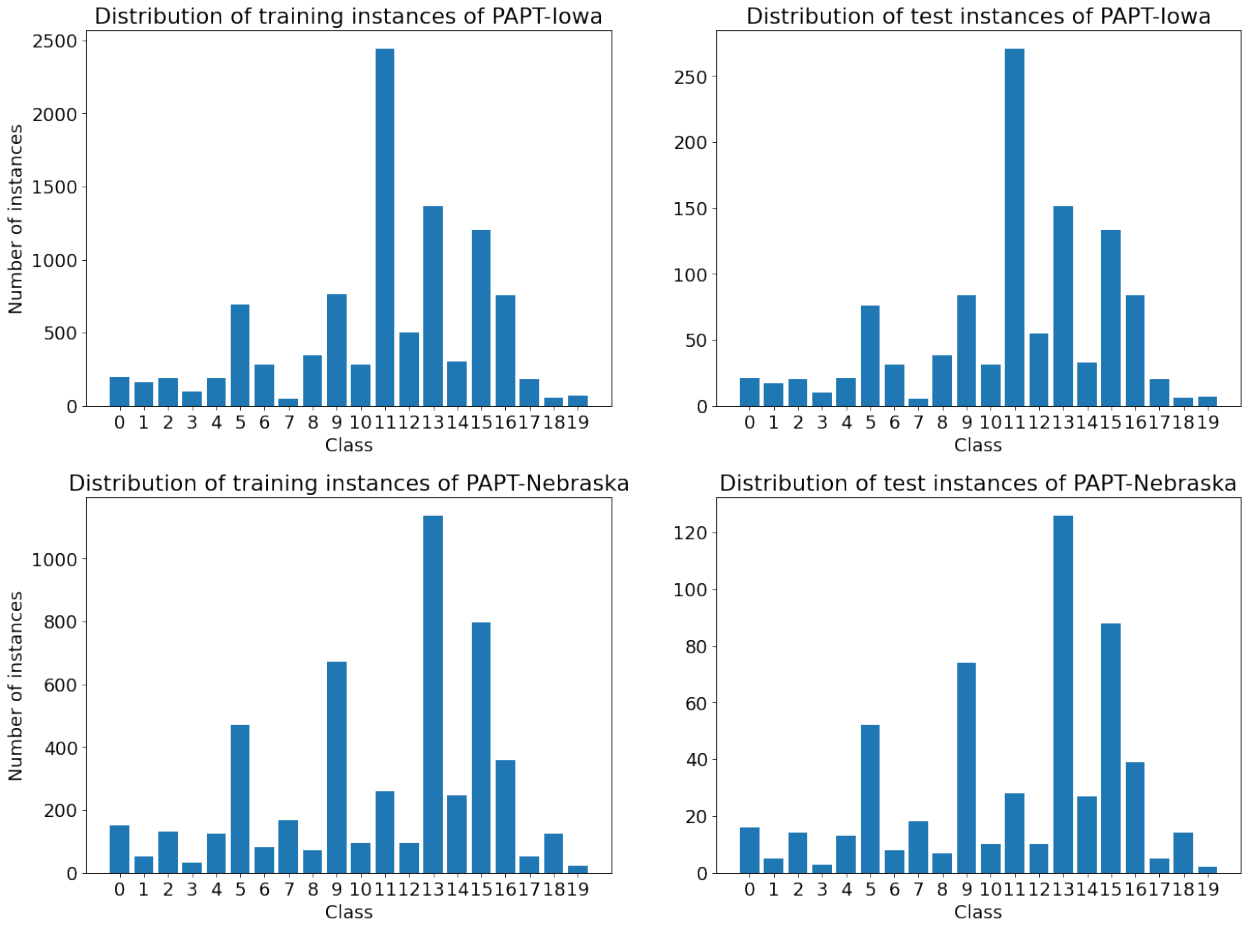


Figure 5.1 Distribution of documents in the training and test datasets of PAPT-Iowa and PAPT-Nebraska.

represent the PAPT dataset specifically from Iowa and Nebraska legislators, respectively. Figure 5.1 shows the training and test data distribution for each class.

5.2 Experimental Settings

The training of both the underlying classifier and QuaNet was done by using the default implementation in the `QuaPy` library [30] available online[1]. This implementation uses a bi-LSTM network for the classifier model with all the same hyperparameters as in [10]. The classifier was

[1] https://github.com/HLT-ISTI/QuaPy

trained to minimize the Cross-Entropy loss for a maximum of 200 epochs with an early stopping criterion. The training loop was ended after no improvement in the validation accuracy was observed for 10 consecutive epochs. No data augmentation was used for any of the datasets. QuaNet was trained with stochastic gradient descent to minimize the Mean Square Error (MSE) of the predictions. A batch size of 100 was used with a maximum number of iterations of 100. The training data was split into three parts, where the first two parts remained as the training data, and the third part was used as the validation data. The same early stopping criterion was also applied in training QuaNet. In [30], the list of document representation and posterior probabilities, L, was sorted based on a given target class for binary quantification tasks, and not sorted for multi-class quantification tasks. We kept this default configuration in obtaining the results in Chapter 5.4 and 5.5, as [10] only tackled binary quantification and did not mention sorting L for multi-class quantification.

5.3 Performance Metrics

We compared the performance of the quantifier using three different performance metrics: Mean Absolute Error (MAE), Relative Mean Absolute Error (RMAE), and Kullback-Leibler Divergence (KLD). Each of them is defined as below:

$$MAE(P, \hat{P}) = \frac{1}{|C|} \sum_{c \in C} |P(c) - \hat{P}(c)| \tag{5.1}$$

$$RMAE(P, \hat{P}) = \frac{1}{|C|} \sum_{c \in C} \frac{|P(c) - \hat{P}(c)|}{P(c)} \tag{5.2}$$

$$KLD(P, \hat{P}) = \sum_{c \in C} P(c) \log \frac{P(c)}{\hat{P}(c)} \tag{5.3}$$

where P is the true class distribution; $\hat{P}$ is the predicted class distribution; $|C|$ is the total number of classes; and $|P(c) - \hat{P}(c)|$ is the absolute value of the difference between the true and the predicted ratios of class c. A small value of error or divergence is desirable.

5.4 QuaNet's Quantification Performance

We first tested QuaNet's performance against the entire test dataset for each of the five datasets. Table 5.2 shows the validation accuracy of the underlying classifiers and QuaNet's quantification performance. The resulting classifiers from the PAPT datasets are due to the highly imbalanced class distributions in these datasets. QuaNet shows the best performance on the YELP dataset and the worst performance on the PAPT datasets. It is also observed that QuaNet shows better performance when the dataset is balanced and its corresponding classifier is more accurate.

Table 5.2 Performance of classifiers on validation dataset and QuaNet's quantification performance on test datasets.

	Classifier	QuaNet		
Dataset	val-acc	MAE	KLD	RMAE
IMDB	82%	0.0253	0.0013	0.0503
YELP	95%	0.0108	0.0002	0.0213
AG-NEWS	85%	0.0221	0.0055	0.0930
PAPT-Iowa	69%	0.0593	0.9459	1.5164
PAPT-Nebraska	62%	0.0470	0.6878	0.9568

5.5 RQ1: What input features are important to the final class distribution prediction of QuaNet?

To find the features and their contribution to the prediction, one method is to give the entire test dataset as input. However, as shown in Figure 5.2, this method only gives one contribution per input without descriptive statistics of the contribution for each input. Alternatively, we created multiple samples to be tested, each with 100 documents, from a given test dataset.

We designed different sampling strategies for different test datasets to reflect the difference in the number of classes and the class imbalance ratios. For binary quantification tasks (IMDB and YELP datasets), we made the prevalence of a positive class for different samples take a value in {0, 0.01, 0.02, ..., 0.99, 1}. We ensured that the class ratios of the positive and the negative

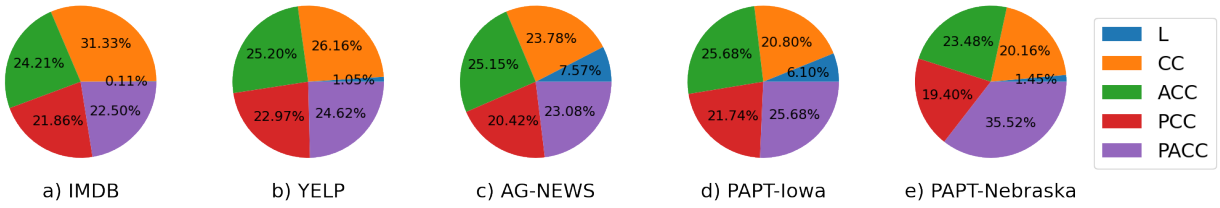
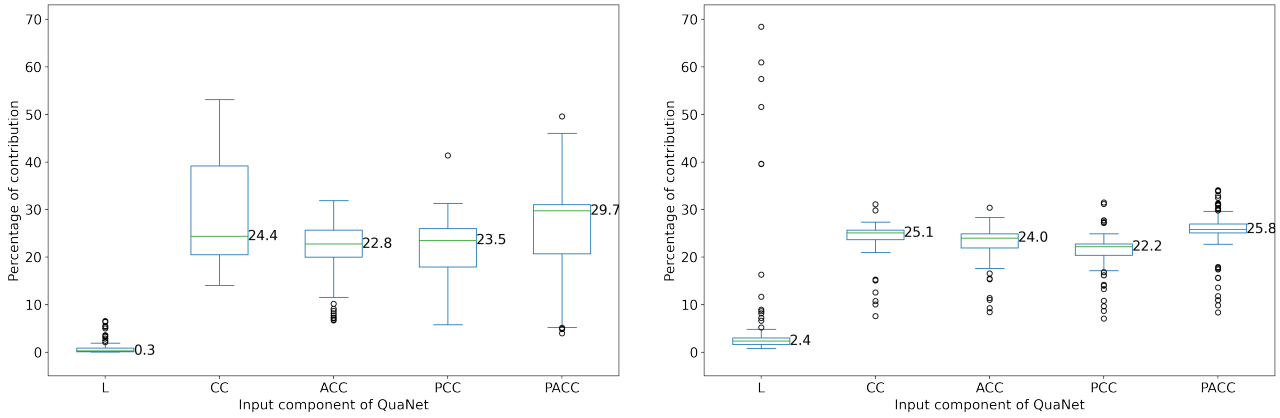


Figure 5.2 Percentage of contributions of each input component towards the final prediction of QuaNet on different test datasets. For each dataset, the contribution of each component is calculated with Equation 5.4.

classes sum to one, e.g. if the positive class ratio is 0.99, then the negative class ratio is 0.01. This strategy created 101 samples with different distributions. For multi-class (four classes) quantification tasks on AG-NEWS, the prevalence of each class took a value in {0, 0.1, 0.2, ..., 0.9, 1}. We ensured that the sum of the four class ratios was one. This strategy resulted in 282 different samples. When assigning documents to a given sample, sampling without replacement was used if the specified prevalence for the class was larger than the actual prevalence of the class. Otherwise, sampling with replacement was used. For twenty-class quantification on both the PAPT datasets, we randomly generated 100 samples instead of restricting each class to a specified class ratio due to the following reasons. 1) To avoid too many combinations of class distributions. 2) To prevent the creation of samples with many duplicate documents caused by sampling with replacement since the PAPT datasets are small and the class distribution is highly imbalanced.

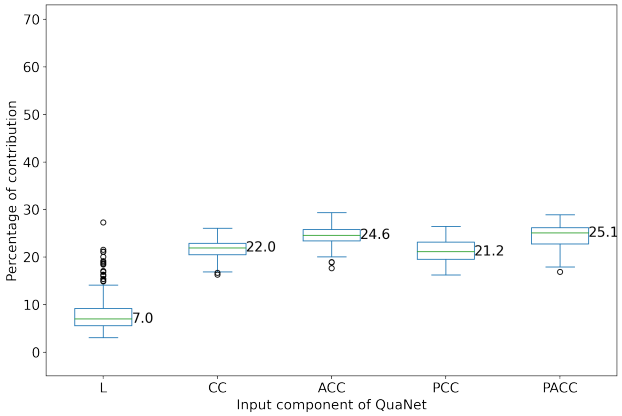
Figure 5.3 shows the boxplot of contributions of each of the five input components. The contributions were calculated from the relevance scores given by Algorithm 1 using Equation 5.4. There are five data points per sample in Figure 5.3 where each of the data points for the sample is the contribution of the corresponding input component shown on the x-axis towards the final prediction of QuaNet. Let R_X denote the sum of the absolute relevance scores for each dimension of the component X. The contribution of X is calculated using Equation 5.4.

$$\textit{contribution of input } X\ (\%) = \frac{\sum R_X}{\sum_{y \in S} R_y} \times 100 \tag{5.4}$$

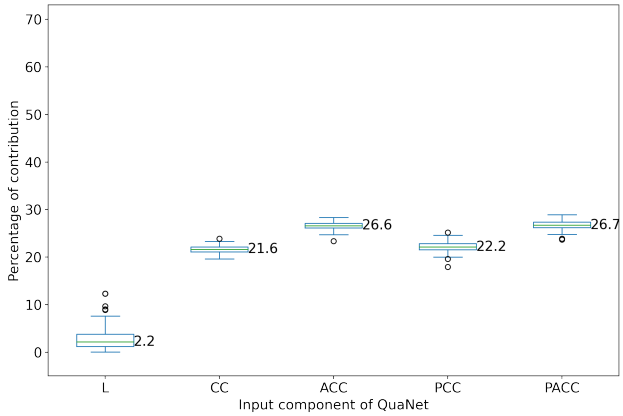


a) IMDB

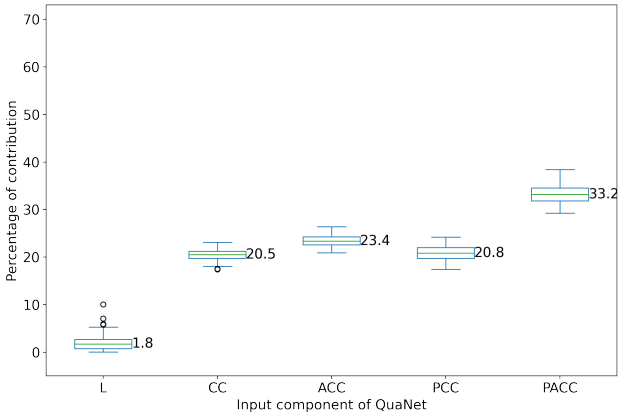
b) YELP



c) AG-NEWS



d) PAPT-Iowa



e) PAPT-Nebraska

Figure 5.3 Percentage of contributions of each input component towards the final prediction of QuaNet on the testing sets of different datasets.

For example, the CC component is a list of $|C|$ numbers, where each number represents the predicted class distribution of a corresponding class. In Algorithm 1, `LRP` returns a relevance score for each of these numbers, thus the total contribution of the CC component is the sum of the absolute relevance scores of all $|C|$ predictions by CC. The contribution of the input is represented as a percentage for better comparison and visualization between results from different samples.

In Figure 5.3, for each input component on the x-axis, the box in the boxplot indicates the center of the distribution of the contributions for that component. The solid line across the box shows the median value. The bottom and the top of the box are drawn at the first quartile and the third quartile, while the whiskers represent 1.5 times of interquartile range away from the first and third quartile, respectively. Any data point outside of this range is considered an outlier. The figure shows that the sorted list L consistently has the lowest contribution towards the final prediction. This component has a median contribution of less than 5% in four out of five of the boxplots. Furthermore, there are more outliers with the public datasets compared to the PAPT datasets. For the outliers shown in Figure 5.3b and c, the component L contributes the most among the input components. Upon a closer investigation, we found that all the outliers in Figure 5.3a, b, and c resulted from samples with highly skewed class prevalence. This shows that QuaNet does rely on L when the sample distribution is highly different than the class distribution of the training dataset. This also explains why the data points in Figure 5.3d and e are highly clustered and have fewer outliers, as the samples were produced randomly. Thus, the class prevalences of different samples are similar to each other. It is interesting to note that the contributions were spread out more in Figure 5.3a compared to Figure 5.3b and c.

The predictions from the four quantifiers, CC, ACC, PCC, and PACC, contribute on average more than 20% respectively, totaling over 90% of the contribution towards QuaNet's decisions. PACC is consistently the highest contributing component, meaning that PACC's prediction is considered the most valuable information in QuaNet's decision. This aligns with the finding in [31], where PACC was found to be the top-performing quantifier among these four quantifiers. Having a total of over 90% contribution from the quantifiers reflects that QuaNet learns to rely

heavily on the results from these quantifiers more than the quantification embedding. To test that the quantifiers' prediction is sufficient, we trained a simple neural network, called Small-NN, to predict the class prevalence with only these four quantifiers' predicted class prevalence as input.

Table 5.3 Performance of QuaNet vs Small-NN. Small-NN is a simple fully connected network that takes the four quantifiers' predicted class prevalence as input. Bold numbers indicate better performance

Model	QuaNet			Small-NN		
Dataset	MAE	KLD	RMAE	MAE	KLD	RMAE
IMDB	0.0907	0.0613	0.2278	**0.0560**	**0.0292**	**0.1686**
YELP	0.0216	**0.0026**	**0.0800**	**0.0179**	0.0273	0.0801
AG-NEWS	0.0370	0.0749	0.3130	**0.0177**	**0.0122**	**0.1224**
PAPT-Iowa	0.0612	1.1222	1.5903	**0.0117**	**0.0626**	**0.3489**
PAPT-Nebraska	0.0514	0.9164	1.1115	**0.0112**	**0.0570**	**0.3361**

Small-NN has one fully connected layer with 64 neurons with ReLU activation, followed by a final layer of $|C|$ neurons with softmax activation. The training and test data for Small-NN were produced by discarding the L input component to QuaNet during the training and testing. 30% of the training data was used as the validation data for training purposes. Small-NN was trained to optimize MAE with Adam optimization at a learning rate of 10^{-3}. It was trained for 50 epochs with the stopping criterion to end the training after no improvement in validation loss for 2 epochs. Table 5.3 shows that Small-NN, a much less sophisticated model, outperformed QuaNet on all the datasets based on MAE. This outcome aligns with our interpretation that the predicted class distributions of the four quantifiers are more valuable as quantification features.

5.6 RQ2: Does sorting L by $\mathbf{Pr}(c|\mathbf{x})$ increase the performance of QuaNet?

Esuli et al [10] proposed to sort the list L because of the intuition that RNN can learn to count by observing the ordered sequence of $\Pr(c|\mathbf{x})$ values, thus recognizing the switch point between classes. This RQ is framed to examine if bi-LSTM can recognize the switch point.

We performed an experiment to compare the performance of QuaNet on the IMDB dataset by sorting L on the test data with two other sorting methods. Method 1 sorts the posterior

probabilities of a class different from the class used in training. Method 2 does not do any sorting of the posterior probabilities. Algorithm 1 was used in this experiment with its inputs S^U produced by the different sorting methods. The MAE provided by QuaNet on the IMDB test dataset is as follows (best to worst): 0.0253 (see Table 5.1) for sorting based on the default class used in QuaNet, 0.0706 for no sorting (Method 2), and 0.4835 for Method 1. This result shows that QuaNet achieved the best performance with sorted L as its input.

One hypothesis is that the trained bi-LSTM layer observes the sorted order in L, i.e., sorted by $Pr(c|x)$. Once the posterior probabilities of the test data are not sorted according to the order it learned (sorted by $\Pr(c'|\mathbf{x})$ where $c' \neq c$ or not sorted at all), L has a different pattern that the quantifier has not seen before, thus affecting its prediction. To confirm this hypothesis, we trained different versions of QuaNet by applying different sorting methods on L from the training data and compared the performance. With each dataset, the same classifier is used to train the different versions of QuaNets. We tested each QuaNets with samples created as described in Chapter 5.5. Table 5.4 and Table 5.5 show the performance.

Table 5.4 Performance of QuaNet models trained on binary quantification tasks with different sorting methods. Averaged over performance on samples created as described in Chapter 5.5. QuaNet denotes the model that was trained with default sorting. QuaNet-unsorted denotes the model trained with unsorted L. QuaNet-revsort denotes the model trained by sorting L on the posterior probabilities of a different class than the default.

Dataset	IMDB			YELP		
Model	MAE	KLD	RMAE	MAE	KLD	RMAE
QuaNet	0.0907	0.0613	0.2278	0.0216	0.0026	0.0800
QuaNet-unsorted	0.0837	0.0509	0.2168	0.0195	0.0026	0.0828
QuaNet-revsort	0.0972	0.0553	0.2442	0.0216	0.0033	0.0792

As mentioned in Chapter 5.2, the default configuration for inducing a multi-class QuaNet does not include sorting L. We ran an experiment to compare the performance of a multi-class QuaNet trained with or without sorting L.

Table 5.5 Performance of QuaNet models trained on multi-class quantification tasks with or without sorting. Averaged over performance on samples created as in Chapter 5.5. QuaNet-sorted denotes the model that was trained with sorting L on the posterior probability of one of the classes. QuaNet-unsorted denotes the model trained with unsorted L, which is the default configuration in [30].

Dataset	AG-NEWS			PAPT-Iowa			PAPT-Nebraska		
Model	MAE	KLD	RMAE	MAE	KLD	RMAE	MAE	KLD	RMAE
QuaNet-sorted	0.0377	0.0651	0.3214	0.0510	1.0872	1.2434	0.0516	0.8162	0.9433
QuaNet-unsorted	0.0370	0.0749	0.3130	0.0612	1.1222	1.5903	0.0514	0.9164	1.1115

Both Table 5.4 and Table 5.5 show that the performance of Quanet with or without sorting L does not differ significantly. We can conclude that sorting L does not necessarily increase the performance of QuaNet.

CHAPTER 6. CONCLUSION AND FUTURE WORK

We present the first attempt at interpreting a deep text quantifier. By using LRP, we can calculate the relevance of each input component to a quantifier's predicted class distribution and can identify important quantification features. This work presents an interpretation of QuaNet, a recent deep-learning quantifier for text quantification tasks. Our findings are as follows. 1) QuaNet relies the most on one of its inputs, the PACC prediction. QuaNet learns that PACC is the most reliable feature to guide its prediction. This finding is consistent with the study by [31], that PACC is the top performing quantifier among CC, ACC, PCC, and PACC. 2) The quantification embedding used in QuaNet [10] is an important quantification feature when the class prevalence of a given test dataset is significantly different from that of the training dataset. 3) On the five test datasets used in this study, a small fully connected network with CC, ACC, PCC, and PACC as input gives a better average quantification performance than QuaNet does.

In our future work, we plan to interpret the other deep text quantifier, DQN. This is where the advantage of LRP on returning signed relevance score should stand out. DQN is a direct quantification model which does not need any classifier. For DQN, LRP can trace the relevance back to each word in a document in the input. This allows word-level interpretation where we can identify both the positive and negatively contributing words to the quantification prediction.

BIBLIOGRAPHY

[1] Ali, A., Schnake, T., Eberle, O., Montavon, G., Müller, K.-R., and Wolf, L. Xai for transformers: Better explanations through conservative propagation. In *ICML* (2022).

[2] Arras, L., Horn, F., Montavon, G., Müller, K.-R., and Samek, W. "what is relevant in a text document?": An interpretable machine learning approach. *PloS one 12*, 8 (2017), e0181142.

[3] Arras, L., Montavon, G., Müller, K.-R., and Samek, W. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Association for Computational Linguistics, pp. 159–168.

[4] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation.

[5] Barranquero, J., Díez, J., and del Coz, J. J. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition 48*, 2.

[6] Bella, A., Ferri, C., Hernández-Orallo, J., and Ramírez-Quintana, M. J. Quantification via probability estimators. IEEE, pp. 737–742.

[7] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological) 39*, 1 (1977), 1–22.

[8] Denil, M., Demiraj, A., and De Freitas, N. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815* (2014).

[9] Endres, D. M., and Schindelin, J. E. A new metric for probability distributions. *IEEE Transactions on Information theory 49*, 7 (2003), 1858–1860.

[10] Esuli, A., Moreo Fernández, A., and Sebastiani, F. A recurrent neural network for sentiment quantification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018*, Association for Computing Machinery. event-place: Torino, Italy.

[11] Esuli, A., and Sebastiani, F. Optimizing text quantifiers for multivariate loss functions. 1–27.

[12] Forman, G. Counting positives accurately despite inaccurate classification. In *Proceedings of the European Conference on Machine Learning*, pp. 564–575.

[13] Forman, G. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 157–166.

[14] González-Castro, V., Alaiz-Rodríguez, R., and Alegre, E. Class distribution estimation based on the hellinger distance. *Information Sciences 218*, 146–164.

[15] Hopkins, D. J., and King, G. A method of automated nonparametric content analysis for social science. *American Journal of Political Science 54*, 1, 229–247.

[16] Jacovi, A., Sar Shalom, O., and Goldberg, Y. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Association for Computational Linguistics, pp. 56–65. event-place: Brussels, Belgium.

[17] Jerzak, C. T., King, G., and Strezhnev, A. An improved method of automated nonparametric content analysis for social science.

[18] Karpathy, A., and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3128–3137.

[19] Khaleel, M., Qi, L., Tavanapong, W., Wong, J., Sukul, A., and Peterson, D. A. M. IDC: quantitative evaluation benchmark of interpretation methods for deep text classification models. *Journal of Big Data 9*, 1, 1–14.

[20] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[21] Law, H., and Deng, J. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 734–750.

[22] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning* (Bejing, China, 22–24 Jun 2014), E. P. Xing and T. Jebara, Eds., vol. 32 of *Proceedings of Machine Learning Research*, PMLR, pp. 1188–1196.

[23] Li, J., Monroe, W., and Jurafsky, D. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220* (2016).

[24] Lim, B., Arik, S. Ö., Loeff, N., and Pfister, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting 37*, 4 (2021), 1748–1764.

[25] Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (2011), pp. 142–150.

[26] Martino, G., Gao, W., and Sebastiani, F. Ordinal text quantification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 937–940.

[27] Milli, L., Monreale, A., Rossetti, G., Giannotti, F., Pedreschi, D., and Sebastiani, F. Quantification trees. In *2013 IEEE 13th International Conference on Data Mining*, pp. 528–536.

[28] Molnar, C. *Interpretable Machine Learning*.

[29] Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning* (2019), 193–209.

[30] Moreo, A., Esuli, A., and Sebastiani, F. QuaPy: A python-based framework for quantification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 4534–4543.

[31] Moreo, A., and Sebastiani, F. Tweet sentiment quantification: An experimental re-evaluation.

[32] Poria, S., Cambria, E., Hazarika, D., Mazumder, N., Zadeh, A., and Morency, L.-P. Multi-level multiple attentions for contextual multimodal sentiment analysis. In *2017 IEEE International Conference on Data Mining (ICDM)* (2017), IEEE, pp. 1033–1038.

[33] Pérez-Gállego, P., Castaño, A., Ramón Quevedo, J., and José del Coz, J. Dynamic ensemble selection for quantification tasks. *Information Fusion 45*, 1–15.

[34] Qi, L. Quantification learning with deep neural networks, 2021.

[35] Qi, L., Khaleel, M., Tavanapong, W., Sukul, A., and Peterson, D. A framework for deep quantification learning. Springer, pp. 232–248.

[36] Qi, L., Li, R., Wong, J., Tavanapong, W., and Peterson, D. A. Social media in state politics: Mining policy agendas topics. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017* (2017), pp. 274–277.

[37] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), pp. 1135–1144.

[38] Saerens, M., Latinne, P., and Decaestecker, C. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. 21–41.

[39] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 618–626.

[40] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[41] Ullah, I., Rios, A., Gala, V., and Mckeever, S. Explaining deep learning models for tabular data using layer-wise relevance propagation. *Applied Sciences 12*, 1 (2021), 136.

[42] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3156–3164.

[43] Wang, S., Huang, M., Deng, Z., et al. Densely connected cnn with multi-scale feature attention for text classification. In *IJCAI* (2018), vol. 2018, pp. 4468–4474.

[44] Xiang, Y., Choi, W., Lin, Y., and Savarese, S. Subcategory-aware convolutional neural networks for object proposals and detection. In *2017 IEEE winter conference on applications of computer vision (WACV)* (2017), IEEE, pp. 924–933.

[45] Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems 28* (2015).

[46] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2921–2929.