5-2023

# Enhancing Prediction Reliability Of Deep Learning By Data Confidence For Recommendation Systems: A Case Study On Named Entity Recognition

Prianka Banik

# ENHANCING PREDICTION RELIABILITY OF DEEP LEARNING BY DATA CONFIDENCE FOR RECOMMENDATION SYSTEMS: A CASE STUDY ON NAMED ENTITY RECOGNITIONA

A Thesis

by

PRIANKA BANIK

Submitted to the Office of Graduate Studies of
Prairie View A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2023

Major Subject: Computer Science

# ENHANCING PREDICTION RELIABILITY OF DEEP LEARNING BY DATA CONFIDENCE FOR RECOMMENDATION SYSTEMS: A CASE STUDY ON NAMED ENTITY RECOGNITION

A Thesis

by

PRIANKA BANIK

Submitted to the Office of Graduate Studies of
Prairie View A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

| | |
|---|---|
| Lin Li<br>Chair of Committee | Xishuang Dong<br>Co-Chair of Committee |
| Lijun Qian<br>Committee Member | Ahmed Ahmed<br>Committee Member |
| Kiranmai Bellam<br>Committee Member | Yonggao Yang<br>Head of Department |
| Pamela H. Obiomon<br>Dean, Roy G. Perry College of Engineering | Tyrone Tanner<br>Dean, Graduate Studies |

May 2023

Major Subject: Computer Science

## ABSTRACT

Enhancing Prediction Reliability of Deep Learning by Data Confidence for Recommendation Systems: A Case Study on Named Entity Recognition
(May 2023)

Prianka Banik, B.S., Bangladesh University of Engineering and Technology
Chair of Advisory Committee: Dr. Lin Li
Co-Chair of Advisory Committee: Dr. Xishuang Dong

Reliability is crucial for industrial recommendation systems. Recent advancement in deep neural networks has greatly improved the performance of modern recommendation systems. However, there is a lack of research on estimating how reliable such recommendation systems are in practical scenarios. Due to the blackbox nature of the deep learning-based systems, many times additional labor has to be involved to examine the prediction accuracy manually, which is costly and time-consuming. To address the problem, we propose a novel approach to estimate the model confidence for a deep learning-based recommendation system. Our approach utilized data statistics to improve the traditional model confidence estimation and maintain the model's high performance. We further proposed a new evaluation metric to properly compare different prediction confidence estimation approaches. Experimental results showed that the external data statistics could effectively improve the prediction reliability by increasing confidence score, which will lead to significant reduction of the time and labors on the system's prediction result examination.



***Index Terms*** — Prediction Reliability, Recommendation Systems, Deep Learning, Data Confidence, Named Entity Recognition

## ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Lin Li and my co-supervisor Dr. Xishuang Dong for their constant support and endless patience in completing my master thesis. I would not have been able to complete my thesis without their guidance in each step of my thesis from beginning to the end. I would also like to thank Dr. Lijun Qian for giving me the valuable opportunity to participate in this challenging but interesting project. His support and valuable suggestions made my thesis more detailed and solid, which greatly enhanced my research skills.

I would also like to thank my committee members, Dr. Ahmed Ahmed and Dr. Kiranmai Bellam for their valuable time and efforts in perfecting my thesis through suggestions and feedback. Besides, I would like to thank my professors in the Department of Computer Science of PVAMU who taught me over the past two years. Their knowledge, caring of students, and devotion to higher education will keep impacting me through my career pursuit.

Furthermore, I would like to thank Ms. Apoorva Banubakode and Dr. Kumar Sricharan from Intuit Inc. which sponsored this research project, for their time and constructive suggestions in my thesis.

Last but not the least, I would like to acknowledge the unconditional love and support of my parents staying overseas and dedicate this thesis to my parents, my husband, Shujon Naha, and my beloved son, Eron Naha.

**TABLE OF CONTENTS**

Page

## LIST OF FIGURES

FIGURE Page

## LIST OF TABLES

TABLE Page

# CHAPTER 1

# INTRODUCTION

Thanks to the advancement of data science in the past decade, especially deep neural networks, machine learning has been applied to many real-world applications. However, a critical understudied aspect about these new machine learning tools is estimating the learning model's prediction reliability. When we use a neural network model for prediction (e.g., classifying specific animals from images), we need to know how confident the model is about the prediction it has just made (e.g., the image contains a dog). This is crucial in industry applications since the learning model may face real world data which is sampled from out of the training distribution. If we have a proper way to find how reliable the model is, we can either accept the prediction result if the confidence is high or intervene if the confidence is low. For classification problems, Softmax probability is traditionally used as the estimate of the confidence or reliability score for the prediction which is not always reliable (we will discuss Softmax activation function in Chapter 2). Thus, in this thesis, we aim at estimating the prediction confidence of deep learning models. In the following sections, we will discuss different aspects of this research[1].

## 1.1 Research Background and Recommendation Systems

This thesis originated from a research project sponsored by Intuit Inc. which expects to improve the machine learning models' prediction reliability for

[1] This thesis follows the style of IEEE.

recommendation systems on the key entities recognized in electronically scanned financial documents. The problem Intuit engineers encountered is that when they applied machine learning models for entity recognition, sometimes prediction score ~~of~~ is high on the classified entity and sometimes the prediction score is low. It is natural for a machine learning model due to its mathematical and optimization-based foundation. However, to ensure the entity recognition quality, extra labors are often required to help examine the classification results when the prediction score is low. This leads to a significant amount of extra time and cost. Thus, how to improve the performance or the prediction confidence of the learning systems becomes the key to the research.

Recommendation systems can be defined as a set of algorithms which can suggest relevant information based on data gathered from user interactions [12]. Such systems can now be found everywhere in the industry (e.g., Amazon, Yelp, etc.) and they are playing important roles in improving customer experiences by providing them with useful and relevant information [13, 14, 15]. Due to the popularity and wide usage of the Internet, companies and individual entities generate massive amount of contents and sharing them with the world. To make most use of these contents, it is crucial to find the most relevant contents based on user's preferences. This is why we are seeing recommendation systems everywhere from social networks to online marketing.

Recommendation systems can be divided into two primary categories, content-based systems and collaborative filtering based systems [12]. The first category, content-based recommendations systems [16, 17, 18], focuses on user personal data and finding relevant items. On the other hand, the collaborative filtering-based systems match other

users information to see the preferences of similar customers to find the relevant item for the target user. Collaborative filtering-based approaches [19, 20, 21] include nearest neighbor-based approaches, machine learning based classifications systems, and data mining techniques ~~etc~~. Some recent approaches have also combined the two approaches to utilize the best of both categories. Also, due to the recent success of deep neural networks, they are now widely used in recommendation systems [22, 23, 24, 25, 26, 27]. Deep neural networks can understand the input data domain all by themselves by extracting rich and semantically meaningful feature representations without any need of hand-crafted features. Also, they tend to work better when a lot of training data is available, which makes them extremely useful for developing recommendation systems [12].

Although many works have been done to improve the effectiveness of recommendation systems, the study of estimating the systems' reliability is still insufficient [28, 29, 30], making this research timely, needed, and important.

### 1.2 Challenges

A major problem of the existing deep learning-based recommendation systems is that they tend to be overconfident in the predictions. This is due to several components in the models and the training pipelines such as Softmax and Batch Normalization. For such classification systems, Softmax is often used as the last activation layer which provides a probability score for the winning class. It is a common practice to use this probability score as a confidence estimation of how reliable the model is about its predictions. However, Softmax sometimes produces high confidence values when the prediction is incorrect or when the data is out of distribution. Similarly, it often produces low confidence values

when the prediction is correct. Due to these issues, the Softmax probability value of the predicted class cannot be considered as a reliable estimation of the model's actual confidence. Understanding how reliable a model predicts is critical in an industrial setting. While developing an automation system, if we can understand when a model is not confident about its predictions, we can intervene and inspect that particular occurrence. This will make the system more reliable and practical for industrial use.

## 1.3 Method

In this thesis, we investigate the problem of estimating model's prediction confidence for deep learning-based classifications systems and propose a novel approach to solve this problem. We started with the traditional approach of using Softmax predictions from a deep neural network based classifier as the confidence estimation and then augmented it by incorporating a novel data statistics term which increased the confidence score when the prediction was correct and similarly lowered the confidence score when the prediction was incorrect.

### *1.3.1 Model Confidence*

Following the common approach of regular deep learning-based classification systems, we also used a Softmax activation function at the end of the model. The Softmax activation function takes an input vector of logistic scores generated by the model which has the same dimension as the number of classes in the dataset. Then the Softmax function generates a probability score for each of the classes by normalizing the exponential of the input values. We considered the class with highest Softmax score as the predicted class and the corresponding Softmax value as the model confidence estimation.

### *1.3.2 Data Confidence*

As we have discussed above, the traditional way of estimating model confidence using Softmax scores may produce unreliable confidence scores. Thus, we introduced an extra term which we call the data confidence and incorporate it with the model confidence to predict the final prediction confidence score. The data confidence term is calculated based on the data statistics of individual items in the dataset. We designed the data confidence term in such a way so that for highly frequent items, we got a lower data confidence score and for less frequent items, we got higher data confidence scores. We used a linear interpolation to combine the model and data confidence scores.

### *1.3.3 NER as a Practical Application*

To show that the proposed approach can provide more reliable estimation of model prediction confidence score, we picked a practical application called, Named Entity Recognition (NER), for verification. NER takes unstructured text data and assigns each text token to a predefined category of named entity. This is useful for industry which is trying to make sense of unstructured text data in the wild using automated systems. It is the core of this Intuit's project and an essential task in natural language processing. Accordingly, we proposed a new evaluation metric to correctly quantify the performance of a confidence estimation system. With extensive experiments, we have shown that the approach performed significantly better than the traditional approach of estimating model confidence for NER task.

## 1.4 Programming Language

All experiments conducted in this research were implemented in Python which is a popular computer programming language for machine learning. Python is open source. Thus, we did not need to buy any commercial license for developing the methods in Python. From the native Python libraries, we have used NumPy for mathematical operations, Panda for reading and writing data files and IPDB for debugging purposes. The approach is based on deep neural networks which the native Python libraries do not support. Thus, we have used PyTorch library as an external deep learning framework [31]. The syntax of PyTorch is similar to NumPy. PyTorch provides easy to develop deep neural networks as well as training and evaluation packages. PyTorch supports CUDA based GPU operations which were essential for training the models swiftly and observe the results.

## 1.5 Thesis Outline

The outline of this thesis is as follows: in Chapter 2, we discuss the existing works on natural language processing, deep neural networks, recommendation systems, confidence score estimation of deep neural networks, and how data statistics can be used to estimate confidence scores. We focused on the relevant technologies and methods which are explored in the study and provided the context and detailed description of these approaches. In Chapter 3, we give a general idea about the proposed approach. We highlight the problems of current approaches for reliability score estimation for deep neural networks and discuss how data statistics-based information can help to further infer a more reliable prediction confidence score. In Chapter 4, we describe the details of the

approach and the models that we have used to combine the deep learning model based prediction confidence and the data statistics based confidence estimation to generate a more reliable confidence estimation for a particular downstream natural language processing task - Named Entity Recognition. In Chapter 5, we discuss the experimental setup, results, and the findings. Finally, Chapter 6 summarizes the contribution of this study and provides useful insights for future work.

# CHAPTER 2

# RELATED WORKS

This chapter briefly introduces the technologies used in our research.

## 2.1 Feature Extraction of Text Data

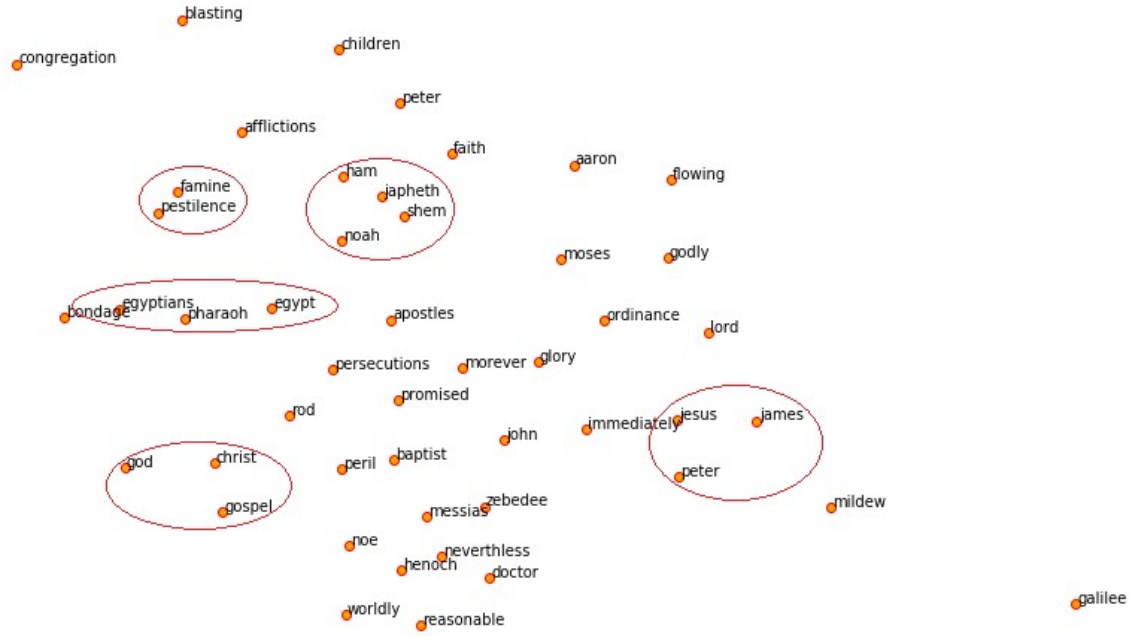


Fig. 2.1. Word2Vec word embedding visualization. Fig. taken from [1]

Extracting semantically meaningful features from training data is the first step for any machine learning system. For text mining and natural language processing tasks, the data is usually unstructured raw text. To infer meaningful information from the raw data, we first need to convert the input to meaningful feature representations. Different aspects of textual features can be extracted for Natural Language Processing (NLP) tasks. The

state-of-the-art and most frequently used methods are word level and character level embeddings.

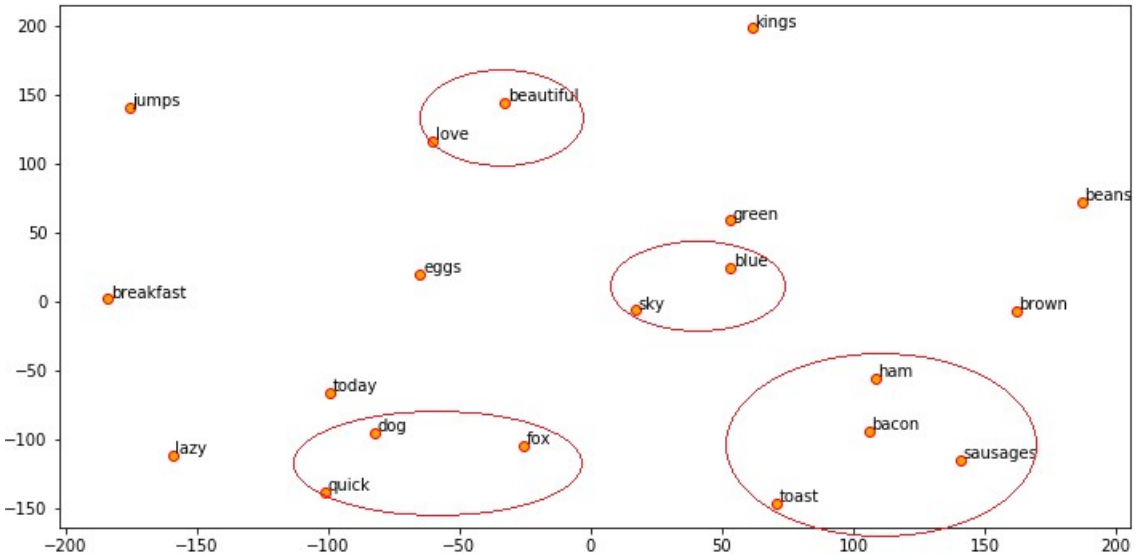


Fig. 2.2. GloVe word embedding visualization. Fig. taken from [2]

### *2.1.1 Word Embedding*

Word embedding is to extract word level tokens from text input and then generating feature representations of these word tokens as the input to the downstream model. The feature representations are usually vectors which are either learned using some machine learning approach or generated using hand crafted mathematical formulas. These feature representations are often referred to as embeddings. Whatever the generation process is, the goals of these embeddings or feature representations are to embed grammatical syntax and semantics. Such mathematical representations of grammatical meaning can provide rich information to perform higher level inference tasks such as part-of-speech recognition. There are several existing machine learning systems for extracting word

embeddings from text data such as Word2Vec [32] and GloVe. Word2Vec utilizes a neural network to either predict a word from its context or predict the context given the target word. The first approach is called continuous bag of words (CBOW) and the second approach is referred to as skip-gram. In this way, the model learns which words are similar in that similar words will have higher vector similarity. GloVe follows a similar approach to Word2Vec but also emphasizes on co-occurrence frequencies of words in the training text data. Both of these approaches learn meaningful vector representations for words which can be seen in the t-SNE (t-distributed Stochastic Neighbor Embedding) plot of the word embeddings, as shown in Fig.s 2.2 and 2.1. In Fig. 2.1, we can see that Noah's sons are all in similar locations, as well as word relevant to Egypt. Similarly, in Fig. 2.2, we can see similar words such as ham, bacon and so on are positioned closely in the embedding space.

### *2.1.2 Character Embedding*

A problem of word level embedding is that if a word is missing in the training data or mis-spelled, it will be impossible to extract feature representations for that particular word. To solve the problem, finer granularity feature representations are learned from characters, which are called character embedding. This embedding contains semantically meaningful feature representation for each word. Character embeddings are learned in similar fashions such as word embedding by learning the contextual representations of each character in the training corpus. Fig. 2.3 shows the t-SNE plot of character embedding. Here we can see the number is grouped in the same location where the special characters are grouped in another location in the embedding space. Learning such rich

embedding is helpful for downstream NLP tasks such as named entity recognition (NER). It is a fine-grained approach to learn meaningful feature representation from text data. As a character embedding model takes characters as input, it can easily handle out-of-distribution and mis-spelled words as well as words which had very low frequency in training data. This is not possible for models which take words as input.

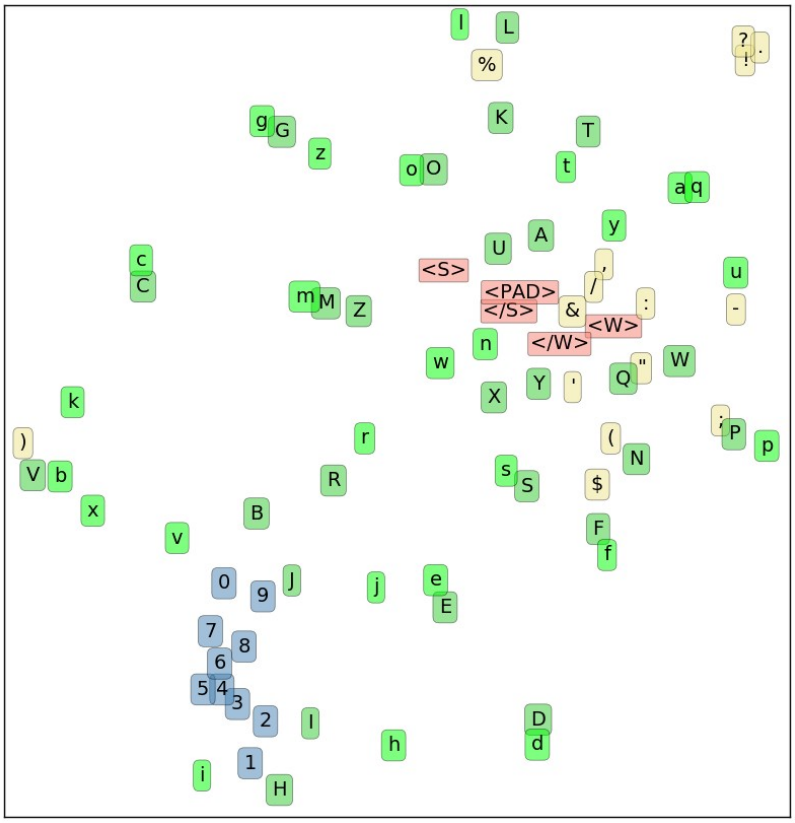


Fig. 2.3. Character embedding visualization. Fig. taken from [3]

## 2.2 Deep Neural Networks

The main algorithms we studied for prediction reliability enhancement is based on deep learning which has been applied to many practical applications today. Over the past decade, a good number of deep neural network architectures were designed, featuring

different feature extraction and problem-solving strengths (e.g., AlexNet, VGG16, GoogLeNET, ResNet, etc.). In the following sections, we will discuss the characteristics of the deep neural networks that we studied in this work.

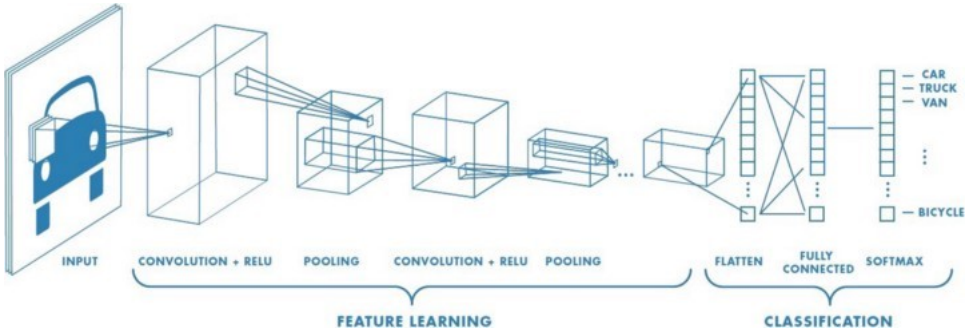


Fig. 2.4. 2D convolutional neural network. Fig. taken from [4]

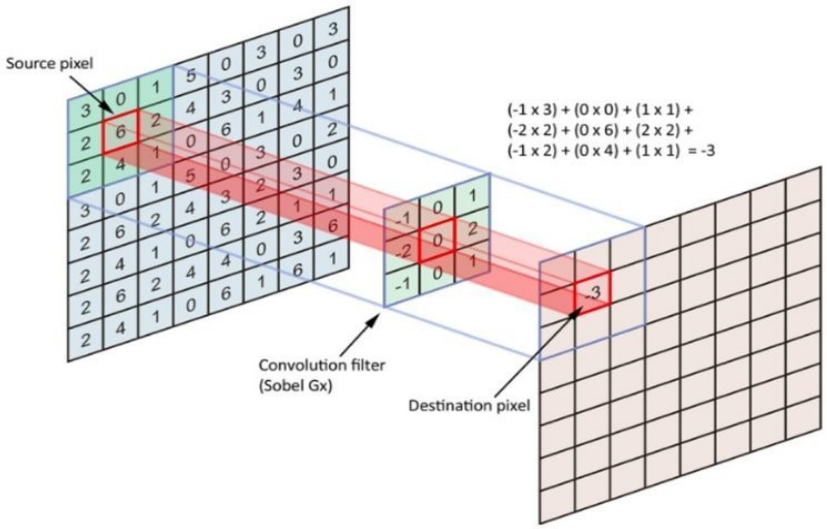


Fig. 2.5. Depiction of 2D convolution. Fig. taken from [5]

### *2.2.1 CNN*

CNN or convolutional neural networks, as illustrated in Fig. 2.4, are specialized models for processing spatial or temporal data. A typical CNN is usually made up of a

series of convolutional layers, followed by ReLU activation function and Maxpooling layers as can be seen in Fig. 2.5. The main part of a CNN model is the convolutional layer which contains learnable kernel matrices which are applied on an input in a sliding window manner to generate the next layer output. These kernels are learned through backpropagation to extract meaningful feature representations from input data. After each convolutional layer, a nonlinear activation function is applied and the most common activation function for CNN is rectified linear unit (ReLU), as depicted in Fig. 2.6. Using ReLU as activation function helps the model to learn non-linearity while also helping to solve the vanishing gradient problem as the activation value after the input greater than zero is unbounded. This provides bigger gradient values to train the model effectively. The next layer used in CNN is the Maxpooling layer as can be seen in Fig. 2.7. Maxpool ensures that the model learns a good summary of the input data while also making the input smaller at every step to reduce the computation burden of the whole model. Usually, dropout layers (not shown in the Fig.) are added into the network architecture to help generalize the model. Finally, for classification CNN models, Softmax activation is used at the end to transform the final layer output to class probability vectors. A depiction of the Sofmax activation function can be seen in Fig. 2.8.

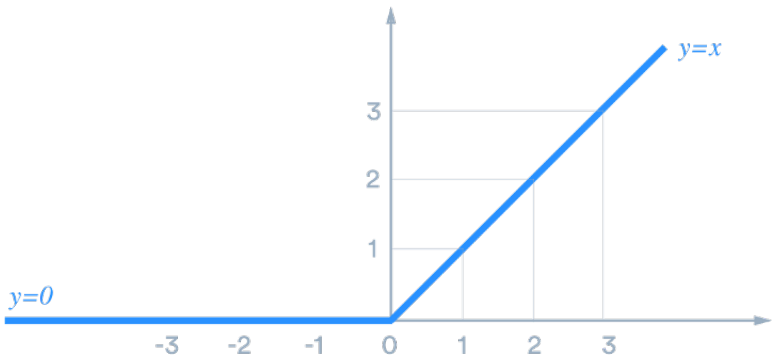

Fig. 2.6. Depiction of 2D ReLU activation function. Fig. taken from [6]

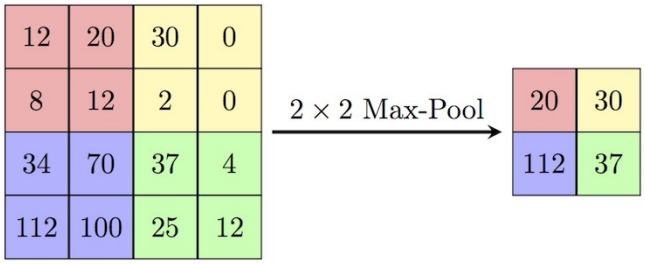


Fig. 2.7. Depiction of 2D maxpooling operation. Fig. taken from [7]

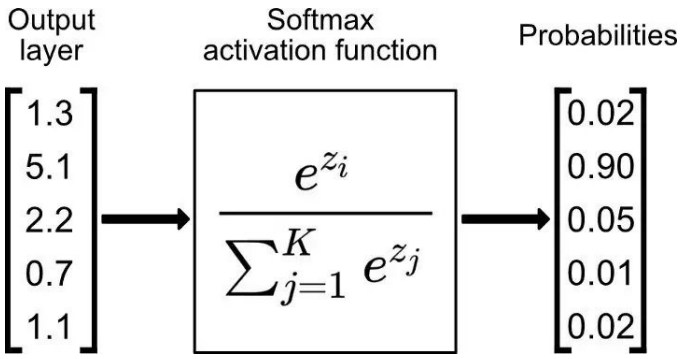


Fig. 2.8. Depiction of the Softmax operation. Fig. taken from [8]

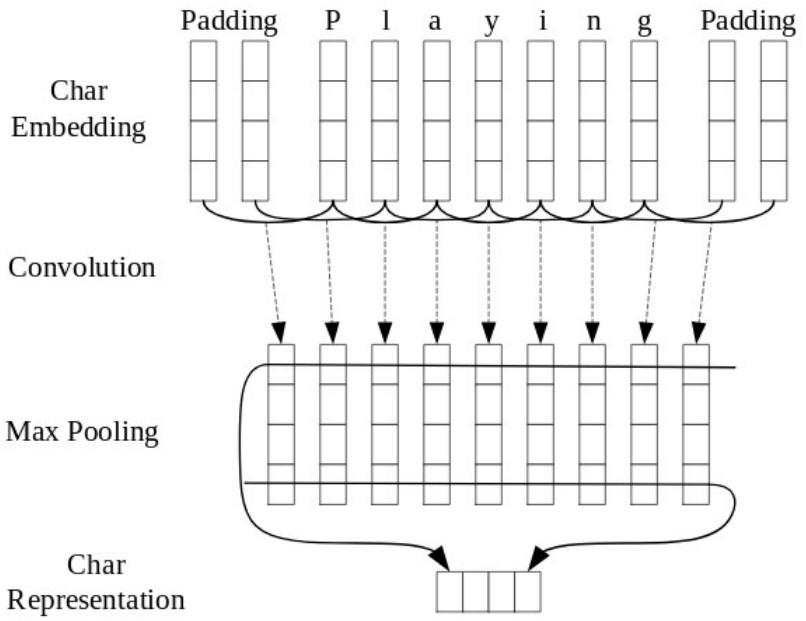


Fig. 2.9. Character CNN model. Fig. taken from [9]

### *2.2.2 Character Embedding CNN Model*

The character embedding CNN [9] model is a particular 1D CNN model which takes character sequences as input and learns a contextual feature representation or embedding for the whole character sequence. Fig. 2.9 shows the character embedding CNN model. Before passing to the model, each character is turned into a one-hot encoded vector by first generating a vector with all zeros and same size as the alphabet (e.g., 26) and then assigning 1 only to the corresponding position of the given character. The convolutional layers contain 1D kernels which slide over the character sequence to generate the next level output and then like regular CNN model max poling and ReLU activation functions are used to learn a single representation for the whole character sequence.

### *2.2.3 LSTM*

LSTM is a form of deep neural network that is specialized in sequential data processing such as text, audio, or video. LSTM solves two crucial problems of regular recurrent neural networks (RNN). The first problem is the long-term dependency where if the input sequence is too long, then regular RNN fails to relate the current term with the previous items in the sequence. The second problem is regarding vanishing gradient where during backpropagation for long input sequences, the gradient is diminished and as a result no real learning can happen. LSTM or long short-term memory networks solve these problems by accommodating various memory gates. There are three gates in LSTM, the input gate, the forget gate, and the output gate. We depict the inner workings of the three gates in a LSTM cell in Fig. 2.10.

The input gate decides how much information from the current input token should be added to the sequence context information. This is useful to include information from the important items of the sequence and ignore relatively irrelevant tokens. The forget gate, on the other hand, decides which information should be discarded from the context to only keep the important information. Finally, the output gate decides how much information from the context and current token should be propagated to the current time step output. The gates can be mathematically expressed as follows,

$$i_t = \sigma(w_i[h_{t-1},x_t] + b_i) \tag{2.1}$$

$$f_t = \sigma(w_f[h_{t-1},x_t] + b_f) \tag{2.2}$$

$$o_t = \sigma(w_o[h_{t-1},x_t] + b_o) \tag{2.3}$$

Here, $x_t$ indicates the current time step input and the $h_{t-1}$ indicates the hidden state output from the previous LSTM cell. $C_{t-1}$ indicates the context vector output from the previous time step. $\sigma$ represents the Sigmoid activation function and *tanh* represents the Tanh activation function. Finally, $w_i$, $w_f$ and $w_o$ refers to the weights of linear layers representing the gates and their corresponding biases are $b_i$, $b_f$ and $b_o$ respectively. $i_t$, $f_t$ and $o_t$ represents current time step output vectors from the input gate, forget gate and output gate respectively. Finally, $h_t$ and $C_t$ indicate the hidden state output and context output from the current time step. By employing these gates, LSTM selectively stores information gathered from the inputs at each time steps and stores a semantically meaningful summary of the whole sequence in the context vector $C_t$. These gates allow LSTM to summarize much longer sequences compared to regular recurrent neural networks. The activation of the forget gate also solves the problem of vanishing gradient during backpropagation by remembering that certain information should not be forgotten. For these reasons, we used the LSTM model instead of regular RNN models in the work.

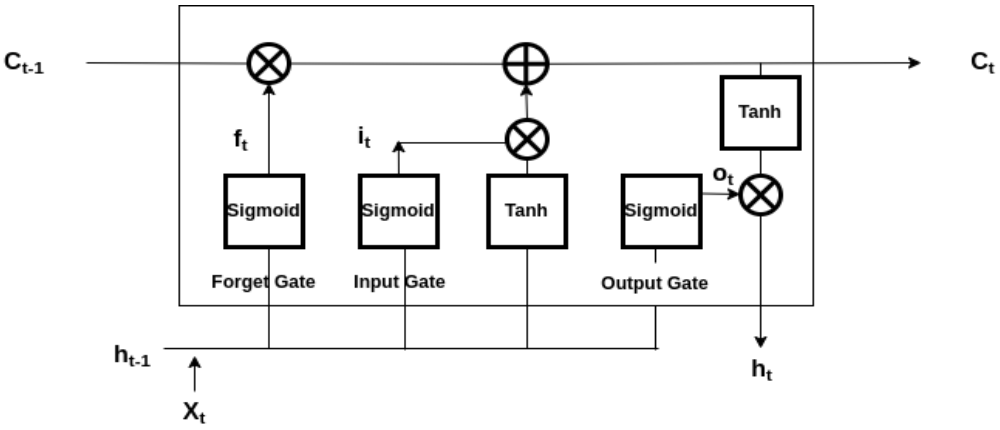


Fig. 2.10. LSTM Memory Cell

### *2.2.4 BERT*

BERT or Bidirectional Encoder Representations from Transformers [33] is a modified transformer model to extract rich discriminative feature representations from input text data. The main task of the transformer is to transform the input sequence to meaningful long-range contextual feature representations, so we can use the learned representations for other downstream tasks such as translation from one language to another (Figure 2.11), sentence classification, NER etc.

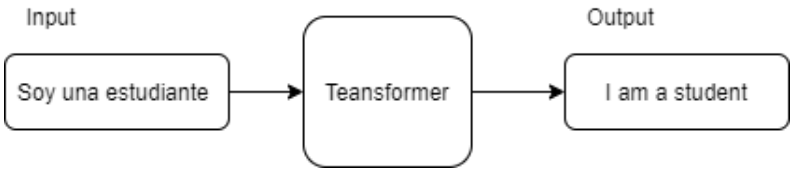


Fig. 2.11. Transformer model. Fig. adapted from [10].

The transformer model is based upon an encoder-decoder architecture where the encoder part consists of different encoder layers which process each input word and produce a feature vector for the corresponding word and then passes it to the decoder layer. The inside of the encoder module is two sub-modules: a self-attention layer and a feed forward layer which is a regular fully connected layer (Fig. 2.12)

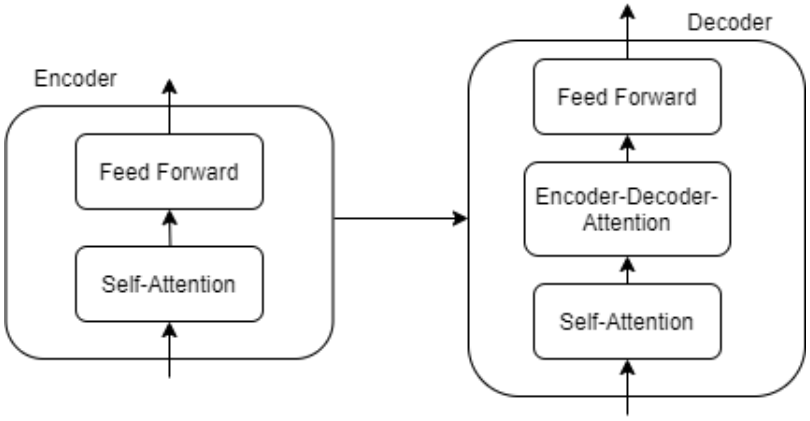


Fig. 2.12. Different layers inside Encoder and Decoder. Fig. adapted from [10]

Each input word first goes through the self-attention layer and then produces the next stage feature vector for the corresponding word. For calculating the feature vector, it considers all other neighboring words of the current input word. Using the relevant contextual words, we can get a more meaningful feature vector for the input word. After the Self-Attention layer, the information of the original word may get lost. That is why there is a residual connection from the original feature vector to the feature vector generated after the Self-Attention layer. After that they are added together and normalized to generate the feature representation for the next encoder layer.

After encoding, the final outputs go to the decoder layer. Decoder network is very similar to the encoder network except it has an extra Encoder-Decoder Attention layer, which decides which encoder output should get higher weights while decoding. The decoder then takes input from the encoder network and can generate a Softmax prediction for each individual word as a sequence of predictions (e.g., NER) or just a single prediction for the entire input sequence (e.g. Sentiment Classification [34]).

BERT uses the transformer model as its base. The model is pre-trained using two self-supervised learning tasks. For the first task, it takes a sentence as an input and randomly masks some words in the sentence. The model's goal is to predict the missing words. For the second task, it takes two sentences and masks random words in both sentences and solves the same problem as before by predicting the missing words. It is better than predicting the next or previous word as those tasks will only provide one directional learning. But masking random words can have bidirectional contextual representation of the input word. These tasks pre-train the model so the model can be used for other downstream tasks like NER.

## 2.3 Deep Learning based Recommendation Systems

Deep learning-based recommendation systems refers to the deep neural network based approaches where the models can extract semantically meaningful feature representation from raw data using only the help of a loss function. The deep learning-based approaches are now widely used in recommendation systems in the industry [15, 35, 25]. Zhang et al. [12] provided a comprehensive list of deep learning-based recommendation systems when they grouped the systems based on the models used and the nature of the input data.

The simplest form of deep neural networks is based on multilayer perceptron networks (MLP). Such models can be used for recommendations systems when the input is in the form of a single vector representation. MLP models take the vector as input and transform it through multiple layers of neurons to generate the recommendation output.

Despite their simplicity, they have been used to process text [36], image [37, 38], audio [39], video [35] and network data [40, 41, 42].

One of the most used deep neural networks is the CNN. CNNs use convolutional kernel-based approach which makes them efficient in terms of number of parameters and inference speed. CNNs are mostly famous for visual data recommendation systems such as images [43, 44, 45, 46, 47, 48, 49, 50, 51] and videos [52] due to their efficiency and capability of learning both local and global information from spatial data. But they have also been successfully used to process audio [18], text [22, 23, 24], and recommendation systems as well.

Another popular deep neural network used in recommendation systems are RNN models. These models are mostly used to process and recommend sequential data such as text data [25, 26, 27]. RNN models take sequence as input and usually use internal memories to learn the context of a large sequence for generating recommendations.

Autoencoders based deep neural networks are also widely used for developing recommendation systems [53, 54, 55]. Autoencoder based models usually contains two parts, encoder and decoder. The encoder takes an input and learns to project it into an embedding space while the decoder takes the embedding as input try to regenerate the input. In this way, the autoencoder model can learn powerful feature representations which can be used for classification and ranking.

Finally, attention-based models have emerged as another popular deep neural network for recommendation systems. They incorporate an extra attention mechanism to focus on only relevant information from input data to make better recommendations. Such

systems showed promising results for both spatial [56] and sequential [22, 57] data recommendation systems.

Overall, none of the above approaches targeted the problem of accurately estimating the confidence of the model prediction which is a crucial part to make any recommendation system reliable and practical for industrial use.

## 2.4 Confidence Score Estimation of Deep Neural Networks

The goal of confidence score estimation approaches for deep neural networks (DNN) is to provide the estimate of uncertainty of the model predictions. The existing works on DNN confidence score estimation can be divided into several categories [58], such as single deterministic methods, Bayesian methods, ensemble methods, and test-time augmentation methods. Single deterministic approaches such as [59, 60] use a separate module to produce the confidence estimation while training the module end-to-end with the original prediction network. Other single deterministic models such as [61, 62] first train the original model and then train a separate module to estimate the uncertainty in the trained model predictions. Bayesian neural network-based approaches are explored in [63, 64]. Ensemble networks-based approaches combine multiple networks to better estimate the model confidence [65, 66]. Finally, test time augmentation-based approaches [67, 68] generate multiple samples from a single test example by performing data augmentation and then get the confidence scores of all these samples to get a better measurement of the prediction uncertainty.

Although the above approaches can often estimate the model prediction confidence more reliably compared with the traditional approaches, none of them explored the idea

of utilizing data statistics to improve the model confidence score estimation approach. In this thesis, we have shown that external data statistics-based approaches can significantly improve the model confidence estimation.

## 2.5 Data Confidence

There have been very few works to utilize data statistics for directly estimating prediction confidence score. Some traditional computer vision-based approaches used pixel data statistics to improve model prediction confidence [69, 70, 71]. In the natural language processing field, TF-IDF features were used in the context of confidence estimation [72]. The most relevant work to the proposed approach is [73]. The authors proposed a new data evidence-based approach to produce an independent confidence estimation which depends on a input term's frequency. But none of these approaches proposed any efficient way to combine model confidence with data confidence to show that they can complement each other and produce a better confidence score.

# CHAPTER 3

# METHODOLOGY

In this chapter, we will present the general idea about the proposed approach. A case study of the proposed approach on a specific NER task for recommendation systems is discussed in Chapter 4.

## 3.1 Approaches to Address the Problem

The Intuit's research problem, as mentioned in Chapter 1.1, can be tackled from two approaches. The first is to design novel machine learning models to achieve better classification performance, including higher accuracy and higher prediction scores on the real entities. Ideally, if a learning model can constantly achieve 100% prediction accuracy, the problem will be solved whether the prediction scores on the entities are low or high. It is the goal of the machine learning, but very challenging for the current statistical and optimization-based learning model design, especially when the dataset is limited or the problem is complicated (e.g., natural language processing). Over the past decade, computer scientists have devised sophisticated deep learning architectures which can often achieve high prediction accuracy, but these models are still far from perfection. Further improving the designs will need the synergy of new datasets, breakthrough of modeling theory, and higher computation power. The second way to deal with the problem is to fully utilize the latest deep learning techniques for prediction but enhancing the distinction of the prediction scores (i.e., increasing the classification confidence of the real entities). This method may not drastically improve the model's prediction performance, but it will

effectively address the problem from the perspective of practical implementation, as we only need to pay attention to the data with low prediction scores, the more the data receiving high prediction confidence, the fewer the extra labors and the less the extra time will be needed for manually examining the prediction accuracy. Therefore, more cost will be saved for the business running.

In this thesis, the efforts focused on the second approach. It should be noted that while we strengthened the classification confidence score, we also tried to improve the learning model's prediction performance (i.e., accuracy) or at least keep the learning model's performance as competitive as those of the state-of-the-art deep learning models.

## 3.2 Learning Model Prediction Confidence

Almost all deep neural network-based classifications systems (e.g., CNN, LSTM, Transformer, etc.) follow a similar pipeline. First, the input is processed by a feature extractor backbone which extracts semantically meaningful feature representations from the given input. The next step is to pass the feature representation through a classifier network, which turns the feature representation into a logit vector with the same size as the number of target classes. These logit values can be any real number. The next step in the pipeline is a Softmax activation function, which transforms the unbound logit values to probability values ranging from zero to one and the sum of all the class probability values is one. The Softmax activation function can be described as follows:

$$\sigma(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ k = 1, 2, \ldots, K \tag{3.1}$$

where $z_k$ indicates the logit value for class $i$ and $K$ is the number of target classes. The class with the maximum Softmax score is considered as the predicted class and the corresponding Softmax score is considered as the model's confidence estimation, i.e., $MC$ for the given instance $w$ is represented by:

$$MC(w) = \max_{1 \leq k \leq K} \sigma(z_k) \tag{3.2}$$

### 3.3 Data Confidence

Besides considering the model's prediction score as the confidence or reliability estimation, we can also use other source of information to further improve the confidence estimation of the predicted results. In the pre-deep learning era, it was common to use hand crafted data statistics-based features for prediction confidence estimation in computer vision model [69, 70, 71]. TF-IDF based approaches have been used in natural language processing tasks for relating confidence estimation with other metrics such as recall and precision [72]. Another common approach in the natural language processing (NLP) field is to consider the rare items with higher importance compared to the items which occur more frequently. This is utilized in [73] to calculate a new data statistics-based confidence estimation, $WordEC(w)$ or the evidence content of a word $w$ in a text corpora, which can be formulated as follows:

$$WordEC(w) = -log(freq(w)/MaxFreq) \tag{3.3}$$

where $freq(w)$ indicates the frequency of the word $w$ and $MaxFreq$ is the frequency of the word with the most number of occurrences. Here a word with high occurrence frequency

will have lower *WordEC*(*w*) value where an item with low occurrence frequency will have higher *WordEC*(*w*) score.

This intuitive approach of estimating confidence score from data statistics can be complimentary to deep neural network-based model confidence scores and be used to enhance the performance of prediction confidence estimation.

### 3.4 Prediction Reliability

As discussed before, a reasonable way to estimate a deep neural network-based classification system's prediction confidence score is to use the Softmax probability of the highest scoring class. But this approach can provide unreliable confidence estimation due to several components in the modern deep neural networks. Calculating confidence scores in this way often produces high confidence scores for incorrect predictions while producing low confidence values for correct predictions. Thus, it is important to investigate new ways to improve the confidence score estimation by correcting the issues of model confidence estimation approach.

### 3.5 Combining Model Confidence with Data Confidence

To solve the drawbacks of the straightforward approach to calculate prediction confidence score, we propose to systematically combine the model confidence term with the data confidence term. We propose a weighted linear interpolation of the data and model confidence to calculate the new confidence score which we name word confidence, $WC(w_i)$,

$$WC(w_i) = \alpha * DC(w_i) + (1 - \alpha) * MC(w_i) \tag{3.4}$$

Here $MC(w_i)$ indicates the Softmax probability score of the model prediction for the input term $w_i$. The intuition is that combining the data confidence term with the model confidence will help to boost the confidences of correct model predictions which would otherwise be lower than a given threshold.

# CHAPTER 4

# CASE STUDY ON NAMED ENTITY RECOGNITION

To evaluate the effectiveness of the proposed method, we applied it to a case study of the Named Entity Recognition (NER) problem. NER refers to the approach to identify the key elements from a given text data (i.e., a sequence of word tokens). The text data can come from electronic documents (e.g., scanned forms) or images which require an extra pre-processing step such as OCR. An illustration of the NER procedure is shown in Fig. 4.1.

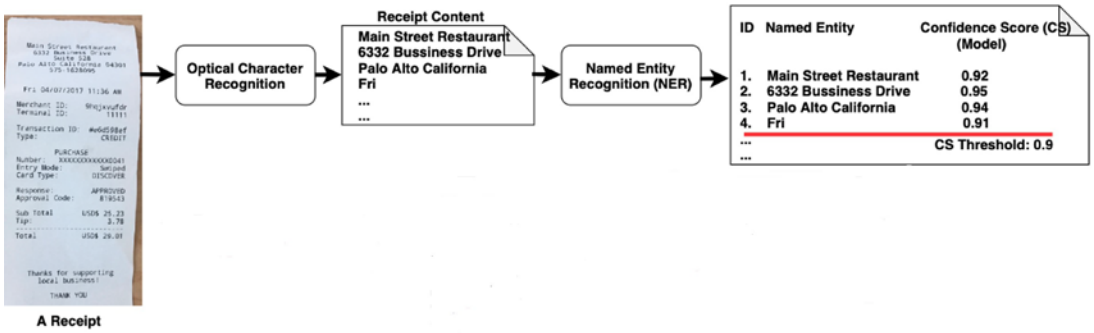


Fig. 4.1. Flowchart of Named Entity Recognition with confidence estimation.

## 4.1 Task Definition: NER Recommendation

There are several practical use cases of NER for natural language processing and biomedical data analyzing tasks. For example, companies like Intuit use NER technology to extract meaningful information from handwritten paper forms to populate their digital versions. Currently deep learning-based models such as LSTM, CNN, BERT, etc. have been successfully used to solve the NER problem. While these models have outperformed

the previous traditional models by a high margin, a major limitation of the current deep learning-based models is that it is often difficult to estimate the reliability of the model's predictions. This is a major issue especially for real world applications where it is important to know how reliable the model prediction is so that humans can intervene in the unreliable prediction scenarios.

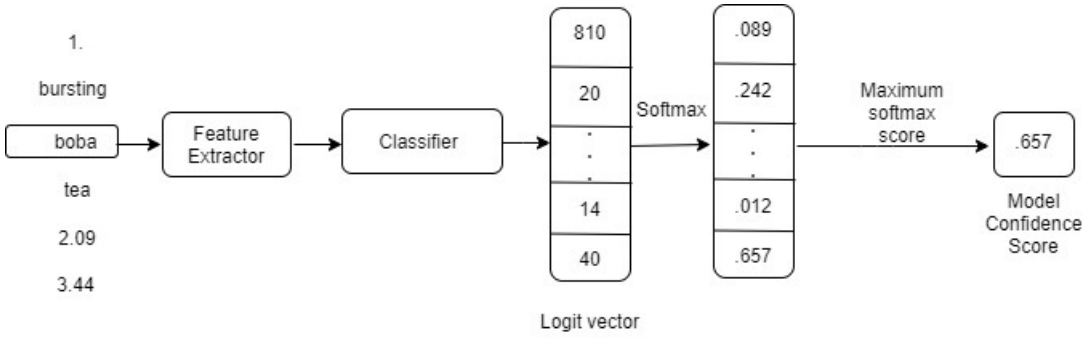


Fig. 4.2. Depiction of NER with confidence estimation

Fig. 4.2 shows the pipeline for a NER system. From the machine learning perspective, NER task can be considered as a multi-class classification task where the classes are the predefined entity categories. Like regular deep learning based multiclass classification models, the common approach for predicting the entity category of a token is to first pass it through a feature extractor following a classifier module. The output of the classifier is a logit vector which has the same size as the number of predefined categories. The logit vector is then passed to a Softmax activation function to generate a Softmax value for each named entity category in the dataset. The class with the maximum Softmax value is considered as the predicted category and the corresponding Softmax value is considered as the estimated model confidence. This is depicted in Fig. 4.2. As

discussed in previous chapters, estimating confidence value in such way can produce unreliable confidence estimation as Softmax function does not provide a well calibrated confidence measurement. Moreover, several other factors of modern deep neural networks such as depth, width, weight decay, and Batch Normalization can also affect the predicted confidence score [74]. In the proposed approach, we fine-tuned the prediction confidence scores generated by deep neural networks for NER prediction task by leveraging data statistics and show that the generated confidence score is a better estimation of the model's true reliability score.

## 4.2 Model Confidence for NER

For the NER task, we have experimented with several deep learning models suitable for the sequence classification problem. In this section, we discuss the different models used to conduct the experiments. All of these models provide Softmax probabilities for the predicted class for each input token which we consider as the model confidence for that particular token. We have used five different models for the NER task, namely CNN, CNN with character embedding, LSTM, LSTM with character embedding [75], and the BERT [33] model. Below we describe these models in detail.

### *4.2.1 CNN*

The CNN model consists of an input word embedding CNN module which takes each word as input and then transforms them into feature vectors, a CNN context encoder to learn rich feature representations from word embeddings, and a multilayer perceptron and Softmax tag decoder module. The detailed architecture of the CNN model can be seen in Fig. 4.3. Each convolutional block in the model consists of a convolutional layer, a

ReLU activation layer, a Dropout layer for regularization, and finally a Batch Normalization layer for 1D data.

The CNN model has six layers. Each convolutional layer in the convolutional blocks contains a 3x1 size kernel with one padding and one stride. The word embeddings are vectors of real values with a length of 300. The first fully connected layer generates a 100-size feature vector and the second fully connected convolutional layer generates 200 size feature vectors.

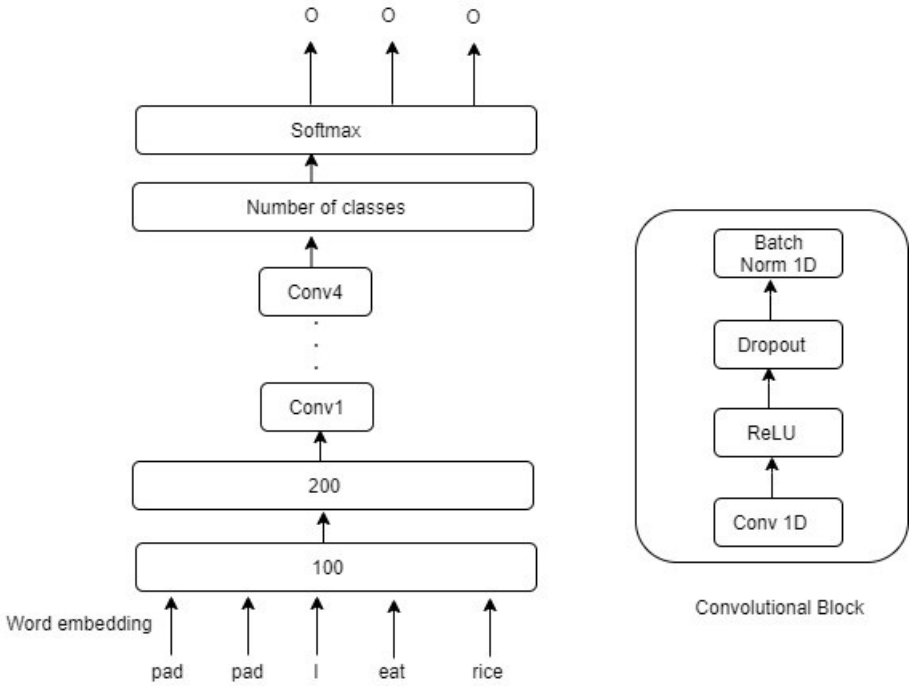


Fig. 4.3. The CNN model

### *4.2.2 CNN with Character Embedding*

The CNN with character embedding [9] model is similar to the CNN model except it uses an extra CNN based character embedding model besides the word embedding model. Fig. 2.9 shows the character embedding CNN model. The model takes the whole

sequence as an input and applies 1D convolution on the character sequence. In this way, we can generate features directly from the character sequence rather than starting from words. The character embedding CNN model has kernel size 3 with padding 1 and stride 1. It takes a series of one-hot encoded vectors each representing a character. The one-hot encoding vector has a length of 27 (26 for the alphabet and 1 for space). The character embedding CNN then converts each of these one-hot encoded vectors to a 50-dimensional feature representation. Fig. 4.4 shows the architecture of the CNN model with character embedding.

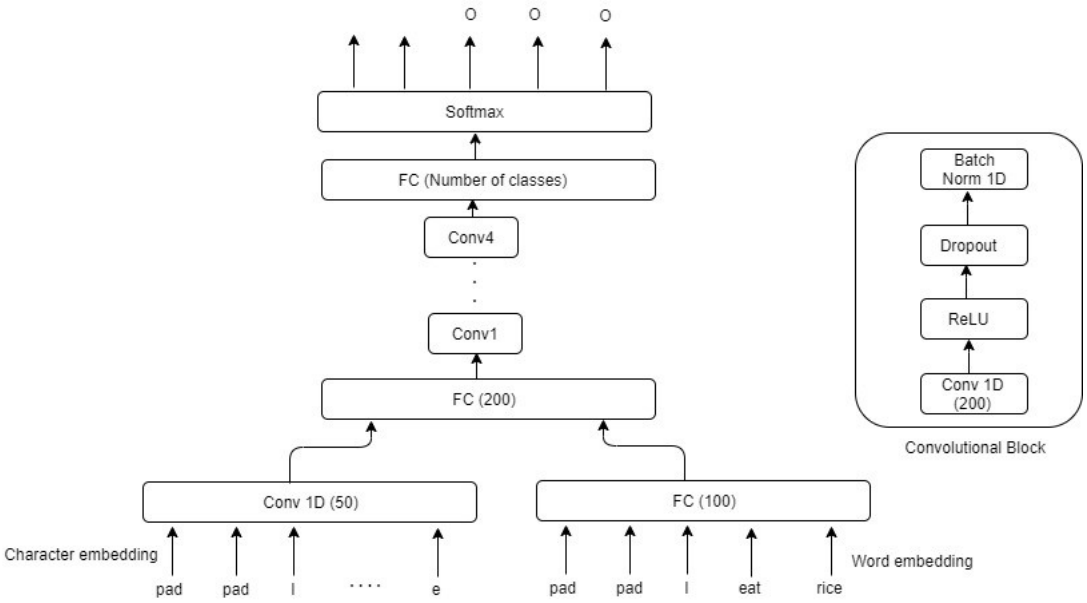


Fig. 4.4. CNN with Character Embedding model

### *4.2.3 LSTM*

The LSTM model used in this research is a bidirectional LSTM [76] for learning input word embedding as well as contextual features representation along with a multi-layer perceptron and Softmax tag decoder. The model is depicted in Fig. 4.5.

The first fully connected layer takes 300-dimensional word embedding and converts them to 100-size feature vectors. Each LSTM cell in both the forward LSTM and backward LSTM model takes 100 size feature vector and again transforms it to a new 100-size feature vector representation.

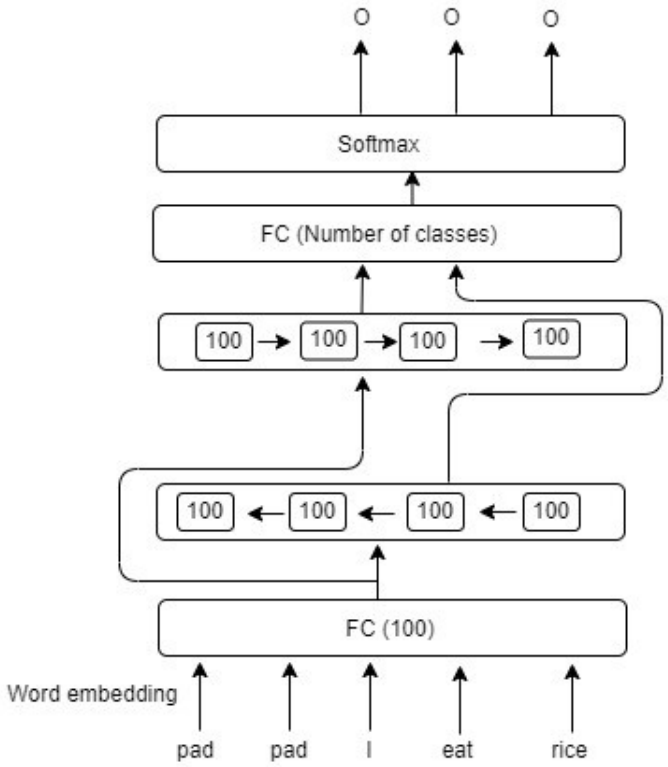


Fig. 4.5. LSTM model

### *4.2.4 LSTM with Character Embedding*

The LSTM with character embedding model is very similar to the LSTM model. Similar to the CNN with Character Embedding model, it uses an extra CNN based character encoding module for input feature learning. The details of this model are shown in Fig. 4.6.

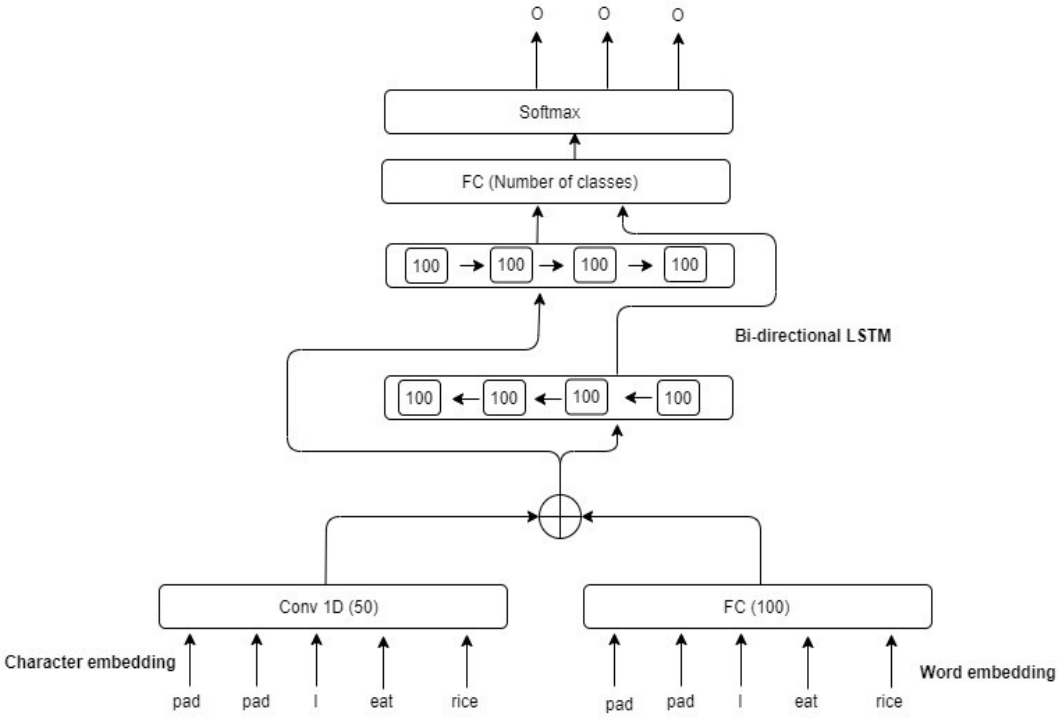


Fig. 4.6. LSTM with Character Embedding model

### *4.2.5 BERT*

In this project, we used the BERT model pretrained on large scale text corpora and then used it as a feature extractor. We then added a multi-layer perceptron with a Softmax layer to build the tag decoder and trained it to generate the tag predictions. The BERT model contains 24 hidden layers and it takes case-sensitive input. The details of this model can be seen in Fig. 4.7.

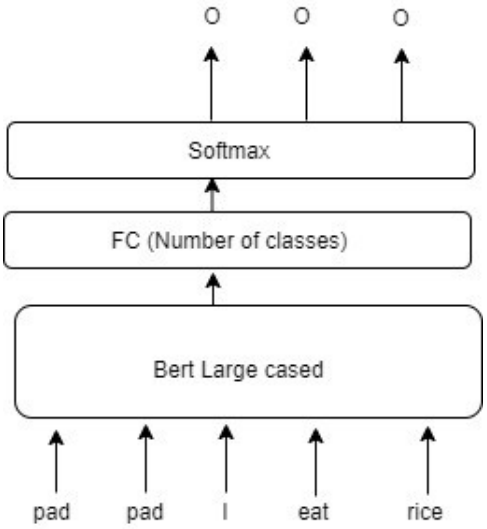


Figure 4.7. BERT model

For all the models, we use the Softmax activation at the end to get the model's confidence. The Softmax activation for the NER task can be described as follows:

$$\sigma_w(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ k = 1, 2, \ldots, K \tag{4.1}$$

where $w$ is a given input token, $z_i$ indicates the logit vector generated for the $i$th entity by the model for token $w$ and $K$ indicates the number of target entities. Then, we can formulate the model confidence for NER for the given token $w$, $MC(w)$ as follows:

$$MC(w) = \max_{1 \le k \le K} \sigma_w(z_k) \tag{4.2}$$

## 4.3 Data Confidence for NER

As for NER, the dataset contains a sequence of words or tokens. We used the following equation to calculate the data confidence of each token in the text corpus.

$$DC(w_i) = -log((freq(w_i) + 1)/freq_{max}) * \beta \tag{4.3}$$

where $freq(w_j)$ indicates the frequency of the $j$th term $w_j$ in the dataset $S$ and $freq_{max} = max_{j \epsilon N} freq(w_j)$, and $\beta$ is the min-max normalization factor. The data confidence value ranges from zero to one. Here, we first find the frequency of each individual term in the dataset and then use it to calculate the corresponding data confidence score so that more frequent words get lower scores, and less frequent words get higher data confidence scores.

### 4.4 Combining Data Confidence with Model Confidence for NER Pre-diction Reliability Estimation

Although we proposed to combine the data confidence with model confidence for enhancing prediction reliability, empirically we found that the straightforward approach as shown in equation 3.4 does not always work well. This is because the data confidence *DC* may not always align with model confidence *MC* and therefore, simply combining them linearly cannot constantly provide the desired confidence estimation. As such, we considered several specific cases and utilized the *MC* and *DC* appropriately for each of these cases. The goal was to get higher confidence, when the model prediction is an 'Entity' and the data confidence was also high. On the other hand, we wanted to reduce the confidence for the cases where the data confidence was low and the model prediction was not an entity as well. Hence, we revised the approach by using *DC* and *MC* to calculate the word confidence *WC*g according to different recognition conditions, which is as follows:

$$\widetilde{WC}(\omega_i) = \begin{cases} \alpha * DC(\omega_i) + (1-\alpha) * MC(\omega_i), & if\ pred_{tag} \in \varepsilon\ \&\ DC(\omega_i) > \rho \\ DC(\omega_i) * \big(1 - MC(\omega_i)\big), & if\ pred_{tag} \notin \varepsilon\ \&\ DC(\omega_i) < \gamma \\ MC(\omega_i), & if\ pred_{tag} \in \varepsilon\ \&\ DC(\omega_i) < \rho \\ 1 - MC(\omega_i), & if\ pred_{tag} \notin \varepsilon\ \&\ DC(\omega_i) > \gamma \end{cases}$$

Where E indicates the set of the classes which belong to the 'Entities' and not to the 'Other' categories, and $\rho$ and $\gamma$ are two thresholds with values in the range [0,1]. In this equation, $\rho$ will be always higher than $\gamma$.

The first case considers the scenario when the data confidence is higher than $\rho$ and the predicted class is an 'Entity'. In this case, we used the linear interpolation as shown in equation 3.4. This is because here we wanted to increase the confidence score so that the confidence for correct predictions gets higher. For the second case, we considered if the prediction is 'Other' and the data confidence is also lower than $\gamma$. Then we first inverse the model confidence by subtracting it from 1 and then further lowered it by multiplying with *DC*. Thus, if we had higher model confidence for non-entities or incorrect predictions, we would lower the confidence score. The third case is that the prediction is an 'Entity' class but the data confidence *DC* was actually smaller than $\rho$. In such a case, we cannot rely on data confidence and simply use the model confidence *MC* as the confidence score. Finally, for the fourth case, we inversed the model confidence, *MC* to compute the final confidence score since the prediction is not 'Entity' and thus either the prediction is incorrect or the actual class for the input is 'Other.'. In that case, we wanted to reduce the confidence score if the model confidence was high (incorrect prediction of 'Other' class) and increase the confidence score if the model confidence was low (correct prediction of 'Other' class).

# CHAPTER 5

# EXPERIMENTAL RESULT ANALYSIS

## 5.1 Datasets

In this research, we used two public datasets: WNUT'17 [77] and GMB [78] to evaluate the proposed method. The datasets contained text data with corresponding BIO (beginning, inside, outside) annotations. BIO annotation is a common format for tagging words in the corpus for several downstream tasks (e.g. Named Entity Recognition, token classification~~, etc~~.). The 'B-prefix' before a tag means the beginning of a chunk, 'I-prefix' before a tag means the inside tag of a chunk, and 'O' means the current tag is ~~a~~ outside tag of a chunk. 'O' tag is used for the words which are not considered as entity or chunk. In the following two sections, we ~~will~~ discuss the details of these two datasets.

### *5.1.1 WNUT'17*

This dataset was introduced by Derczynski et al. [77]. In WNUT'17, the training data was taken from around 1000 annotated tweets. The development data was taken from YouTube and the test data split was drawn from Twitter, YouTube, Reddit and Stack Exchange. Different sources were chosen so that it contained not only noisy user-generated data, but also included longer text up to 140 characters with different writing styles and characteristics. The whole dataset was annotated using six entity types as can be seen in Table 5.1. After applying BIO annotation format, there were a total of 13 NER tags in the dataset including the 'Other' or 'O' tag.

TABLE 5.1. BIO FORMAT OF ENTITY TYPES IN THE WNUT'17 DATASET

| | Categories | | | | | | Total |
|---|---|---|---|---|---|---|---|
| NER type | Person | Location | Product | Creative-work | Group | Other | 6 |
| BIO format | B-person | B-location | B-product | B-creative-work | B-group | O | 11 |
| | I-person | I-location | I-product | I-creative-work | I-group | | |

TABLE 5.2. BIO FORMAT OF ENTITY TYPES IN THE GMB DATASET

| | Categories | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|
| NER type | Geographical Entity | Organization | Person | Geopolitical Entity | Time indicator | Artifact | Event | Natural Phenomenon | Other | 9 |
| BIO format | B-geo | B-org | B-per | B-gpe | B-tim | B-art | B-eve | B-nat | O | 17 |
| | I-geo | I-org | I-per | I-gpe | I-tim | I-art | I-eve | I-nat | | |

### *5.1.2* GMB

Another large-scale NER dataset that was used in this project was the 'Groningen Meaning Bank' or GMB dataset [78]. The contents of this dataset were taken from Voice of America (VOA) together with documents from the MASC dataset and the CIA World Factbook. The updated version of the dataset was released on May 14, 2020. This dataset can be used for named entity recognition or Part-of-Speech (POS) tagging, as well as to generate new text features. A total of eight entities were used to annotate the whole dataset, leading to 17 NER classes including 'Other' after using BIO annotation. A list of the 17 NER classes are shown in Table 5.2

## 5.2 Evaluation Metrics

To evaluate the proposed approach, we designed a new metric called Total Entity

Coverage (TEC) for measuring the ratio of instances with high confidence scores. Besides that, we also used confusion matrices to visualize the prediction performance on individual entity classification.

### *5.2.1 Total Entity Coverage(TEC)*

As explained in Chapter 3, the goal of this research was to produce a high confidence score when the input token is an 'Entity' and the model's entity prediction is correct. To evaluate the confidence scores estimated by the model, we introduced a new metric–TEC. TEC produces higher value when the predicted confidence score is higher than a predefined threshold and the prediction is a correct entity. To better describe TEC, we first introduce the following three terms:

$t$: Threshold of prediction confidence $c[i]$: Confidence of prediction for $i$th token

$S_t$: Set of predictions with confidence higher than $t$

Now we can formulate TEC for a given entity as follows:

$$TEC_{entity} = \frac{N_c}{N_e} \tag{5.1}$$

where $N_c$ is the number of samples in $S_t$ whose predictions are entities, and the predictions are correct; and $N_e$ is the number of samples whose ground truth is entity in the entire test dataset.

We can further describe $N_e$ as follows: $N_e$ = the number of samples in $S_t$ whose predictions are correct + samples in $S_t$ whose predictions are incorrect + samples that are not in $S_t$ but the predictions are correct + samples that are not in $S_t$ and the predictions are incorrect.

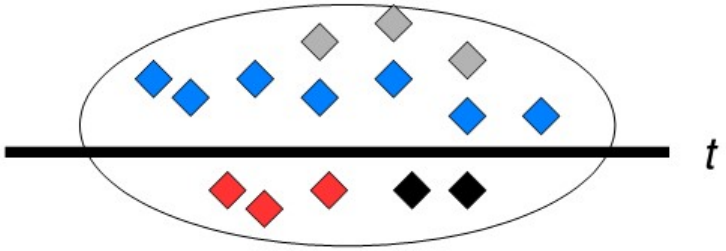


Fig. 5.1. Total entity coverage (TEC)

Fig. 5.1 shows the concept of TEC. Here *t* is the threshold of prediction confidence. The diamonds in the Fig. depicts the entity tokens. Specifically, the blue diamonds are the samples in $S_t$ whose entity predictions are correct and the corresponding confidence scores are higher than the threshold *t*. The grey diamonds are the samples in $S_t$ whose predictions are incorrect, but the confidence scores are higher than the threshold *t*. The red diamonds are the samples that are not in $S_t$ but having correct entity predictions. Finally, the black diamonds indicate the samples that are not in $S_t$ and the corresponding entity predictions are incorrect.

Larger $TEC_{entity}$ value means that more tokens with correct entity predictions are scored above the confidence threshold *t*. On the other hand, lower TEC score indicates either correct entity predictions got lower confidence scores, or incorrect predictions got higher confidence scores. The goal was to generate more accurate confidence estimation so that we can achieve higher TEC value than the baseline approach.

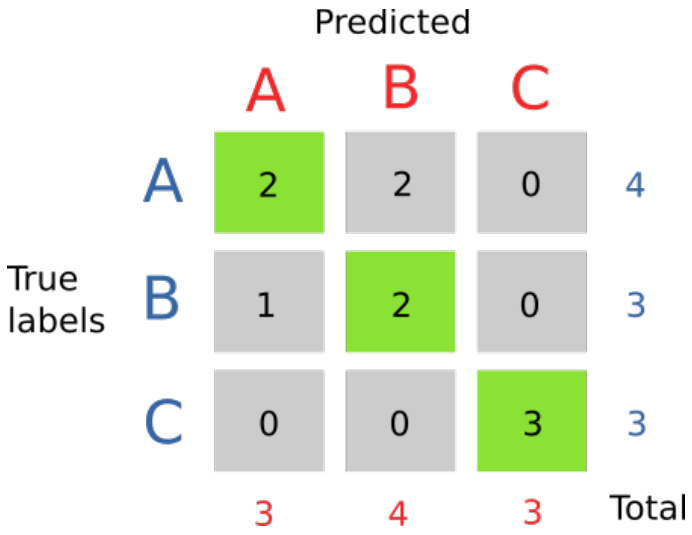


Fig. 5.2. Confusion matrix for multi-class classification. Fig. taken from [11]

### *5.2.2 Confusion Matrix*

Both WNUT'17 and GMB contain multiple entity categories. Thus, we used Confusion Matrix to visualize the performance of the approach and the baseline approach for each category. An example confusion matrix for multi-class classification problem is shown in Fig. 5.2. Each entry of the matrix represents the number of instances that the model predicted with regard to the class marked by the column heading.

The row heading of an entry indicates the ground truth class of the instance(s). The entries in the matrix are color-coded based on their prediction correctness (i.e., correct predictions are in green and incorrect predictions are in gray).

## 5.3 Coding and Experiment Settings

All experiments were conducted on a single NVIDIA V100 TENSOR CORE GPU. Besides using the pre-developed libraries provided by Python and PyTorch, we leveraged

peer researchers' latest work and their source code shared through GitHub to improve the implementation efficiency.

For training the CNN and LSTM models, we used a learning rate of 0.015 and batch size 10. Each model was trained 100 epochs. For optimization we used stochastic gradient descent optimizer provided by the PyTorch library. The CNN, LSTM, and the character embedding models were directly adapted from [79]. For training the BERT model we used a learning rate of $5e^{-5}$ and batch size 32. We trained the model for 100 epochs and for optimization we ~~have~~ used Adam optimizer. The BERT Large-cased model was adapted from Hugging Face PyTorch Transformer library [80]. A code snippet depicting the proposed model is shown in Fig. 5.3.

```
383
384                 if pred_label > other and test_data_conf > rho:
385                     temp_conf = test_data_conf * alpha + pred_conf * (1 - alpha)
386                     new_conf_lst.append(temp_conf)
387                 elif pred_label == other and test_data_conf < gamma:
388
389                     temp_conf = test_data_conf * (1-pred_conf)
390                     new_conf_lst.append(temp_conf)
391                 else:
392                     if pred_label == other:
393                         new_conf_lst.append(1-pred_conf)
394                     else:
395
396                         new_conf_lst.append(pred_conf)
397
```

Fig. 5.3. Code snippet of the linear interpolation of model and data confidence.

For evaluation, all the experiments were executed five times and the average was reported in the results. For inference latency, the approach adds negligible (less than a millisecond) overhead by adding the data confidence term with the model confidence, making it suitable for practical use cases.

## 5.4 Result Analysis

In this section, we present the experimental results and the findings by applying the proposed approach on the two datasets WNUT'17 and GMB. To evaluate the roles played by the model confidence and the data confidence in the combined approach, we used three different $\alpha$ values: 0.95, 0.9, and 0.85 and two thresholds, 0.85 and 0.9 to calculate TEC. We considered the model-confidence-only based approach as the baseline and compared its results with the proposed approach where we calculated an improved confidence score by combining the model and data confidence.

### *5.4.1 TEC Performance and Comparison*

Table 5.3 shows the TEC value for five models, CNN, CNN with character embedding, LSTM, LSTM with character embedding and BERT. As shown in the Table, the approach performs significantly better than the baseline - model confidence based approach in terms of TEC score (higher is better) for all the models. The biggest improvement can be seen for the CNN with Character embedding model where the approach achieved 44 percent improvement over the baseline approach. The optimum confidence threshold was 0.85 as this value consistently gave better scores for all the models. We ~~have~~ also found that the proposed approach was sensitive to the $\alpha$ value and the best $\alpha$ value for all the models was 0.95. Similarly, we have empirically found the best value for $\rho$ is 0.85 and for $\gamma$ the value is 0.2. One interesting observation from Table 5.3 is that for the baseline approach, BERT produced the highest TEC. For the approach, however, LSTM with character embedding model produces the highest TEC value, even though the difference between BERT and LSTM with character embedding was minor.

TABLE 5.3. TEC VALUES FOR WNUT17 DATASET WITH $\rho$ = 0.85 AND $\gamma$ = 0.2

| Model | Threshold | Model Confidence | Data + Model Confidence | | |
|---|---|---|---|---|---|
| | | | $\alpha$ = 0.95 | $\alpha$ = 0.9 | $\alpha$ = 0.85 |
| CNN | 0.85 | 0.222 | **0.338** | 0.336 | 0.329 |
| | 0.9 | 0.2 | 0.308 | 0.304 | 0.293 |
| CNN + Character embedding | 0.85 | 0.266 | **0.384** | 0.384 | 0.381 |
| | 0.9 | 0.233 | 0.353 | 0.349 | 0.340 |
| LSTM | 0.85 | 0.333 | **0.387** | 0.387 | 0.387 |
| | 0.9 | 0.313 | 0.363 | 0.362 | 0.361 |
| LSTM + Character embedding | 0.85 | 0.302 | **0.410** | 0.409 | 0.408 |
| | 0.9 | 0.278 | 0.384 | 0.382 | 0.381 |
| BERT | 0.85 | 0.377 | **0.409** | 0.409 | 0.408 |
| | 0.9 | 0.374 | 0.405 | 0.404 | 0.404 |

Table 5.4 shows the TEC values for five models CNN, CNN with character embedding, LSTM, LSTM with character embedding, and BERT model for the GMB dataset. Similar to the WNUT'17 dataset, the approach again performs better than the baseline approach. The optimal confidence score threshold and the $\alpha$ value are consistent with the Wnut'17 dataset. One interesting finding from this Table is that the difference between the baseline model and the proposed approach is lower for the GMB dataset compared to the WNUT'17 dataset. This is due to the inherent complexities in the WNUT'17 dataset as it was collected from highly unstructured social media data and it was difficult for the models to make accurate predictions. Thus, the approach of combining data confidence with model confidence can significantly boost the performance by complementing the model confidence score. On the other hand, even though the GMB dataset is a much larger dataset compared to WNUT'17, it was still considerably easier and more structured than the WNUT'17 dataset, which made it easier for the deep learning

models to learn the patterns from the data and make accurate predictions. Thus, the improvement of the approach on the dataset is not that significant.

TABLE 5.4. TEC VALUES FOR GMB DATASET WITH $\rho$ = 0.85 AND $\gamma$ = 0.2

| Model | Threshold | Model Confidence | Data + Model Confidence | | |
|---|---|---|---|---|---|
| | | | $\alpha$ = 0.95 | $\alpha$ = 0.9 | $\alpha$ = 0.85 |
| CNN | 0.85 | 0.767 | **0.787** | 0.786 | 0.785 |
| | 0.9 | 0.736 | 0.738 | 0.737 | 0.740 |
| CNN + Character embedding | 0.85 | 0.747 | **0.767** | 0.766 | 0.766 |
| | 0.9 | 0.718 | 0.719 | 0.718 | 0.721 |
| LSTM | 0.85 | 0.764 | **0.781** | 0.780 | 0.780 |
| | 0.9 | 0.742 | 0.739 | 0.738 | 0.742 |
| LSTM + Character embedding | 0.85 | 0.762 | **0.783** | 0.782 | 0.781 |
| | 0.9 | 0.734 | 0.738 | 0.738 | 0.741 |
| BERT | 0.85 | 0.818 | **0.821** | 0.821 | 0.820 |
| | 0.9 | 0.806 | 0.792 | 0.796 | 0.797 |

Fig. 5.4 and Fig. 5.5 present the confusion matrices of the baseline approach and the proposed approach on WNUT17 dataset using LSTM with character embedding model. The parameters used in the approaches are: $\alpha$ = 0.95 and *threshold* = 0.85. Please note that since the focus in this research was to enhance the prediction scores of the entities, the matrices presented here were different from the traditional confusion matrices in that the pictures visualize how many instances in the testing dataset successfully gain the high prediction scores (i.e., greater than or equal to *threshold*) with regard to individual classes rather than the prediction results of the whole testing dataset across the classes. As we can see from the Figs., the proposed approach performed better than the baseline approach in terms of this measurement.

### *5.4.2 Confusion Matrix Performance and Comparison*

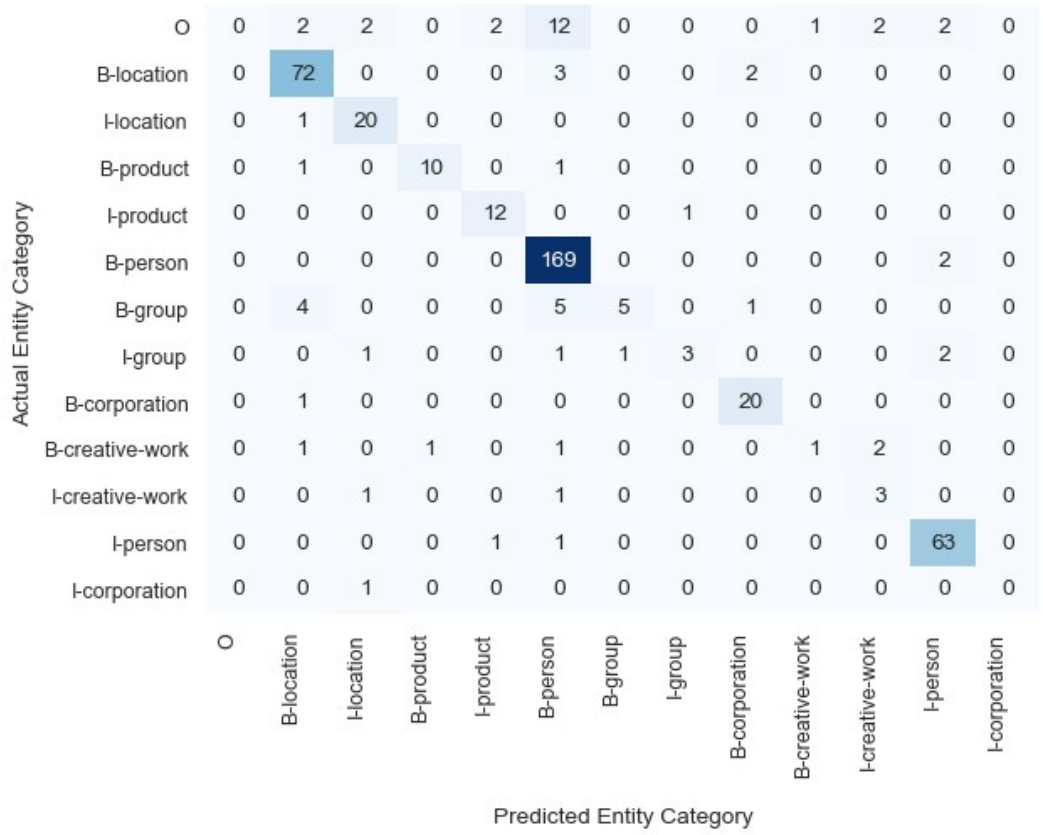


Fig. 5.4. Confusion matrix of LSTM with character embedding model for the baseline approach (model confidence only)

For example, Fig. 5.4 shows that in the baseline approach, the total correct predictions for class ‘B-person’ is 169. After combining model confidence with the data confidence (Fig. 5.5), the correct predictions for class ‘B-person’ increased from 169 to 207. From the results, we can also see that while the model enhanced the predictions confidence of more instances, extra errors (i.e., the entries off the diagonal) also incurred among the instances receiving high prediction scores. Nonetheless, the model still

maintained very competitive prediction accuracy. From the perspective of practical application, this actually leads to another interesting factor to be considered: the trade-off between the cost saved on labor and time for manual examination and the cost incurred due to the extra misclassifications. More comprehensive evaluation metrics can be designed by adding the cost parameter.

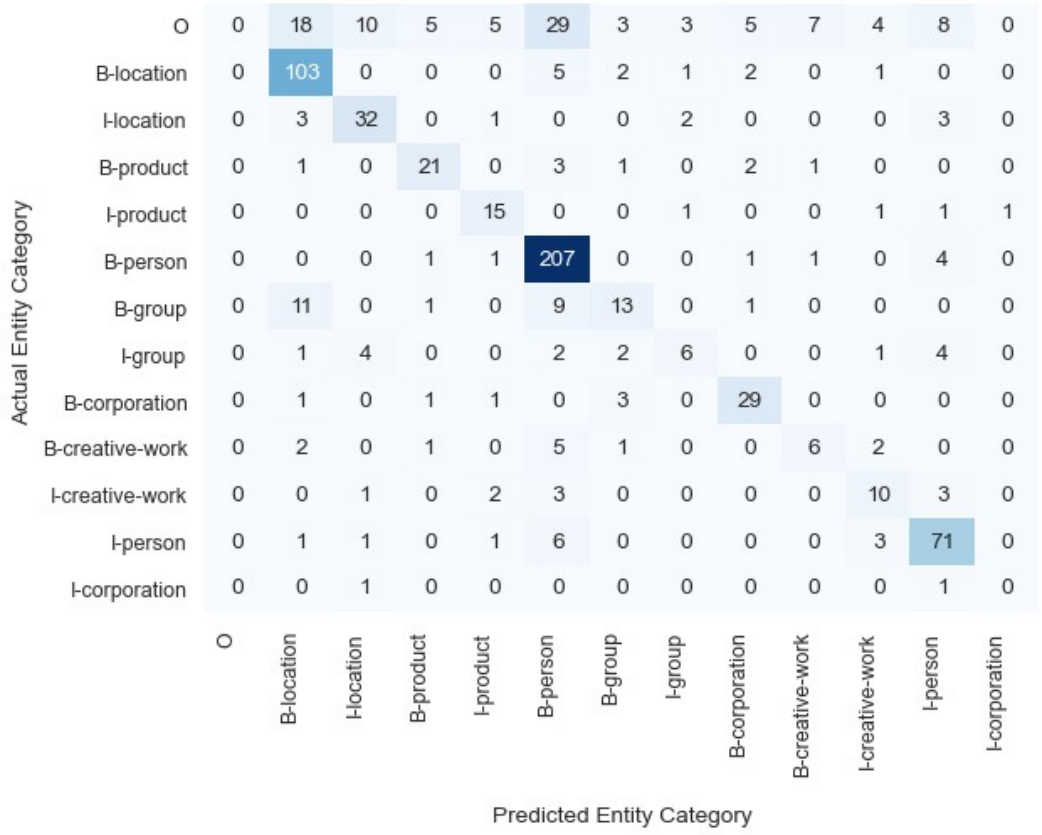


Fig. 5.5. Confusion matrix of LSTM with character embedding model for the proposed approach (model confidence + data confidence)

Fig. 5.6 and Fig. 5.7 show the confusion matrices for the baseline approach and the proposed approach on the overall best performing model BERT on the GMB dataset. Again, matrices present how many instances in the testing dataset successfully gain the

high prediction scores with regard to individual classes. The parameters used in the models were: $\alpha$ = 0.95 and *threshold* = 0.85. Similar to the WNUT'17 dataset, the approach again performed better than the baseline approach in terms of entities receiving enhanced prediction scores. For example, Fig. 5.6 shows that for the baseline approach, the total correct predictions for class 'B-geo' is 6313. After combining model confidence with the data confidence, the correct predictions for class 'B-geo' increased from 6313 to 6324, as shown in Fig. 5.7.

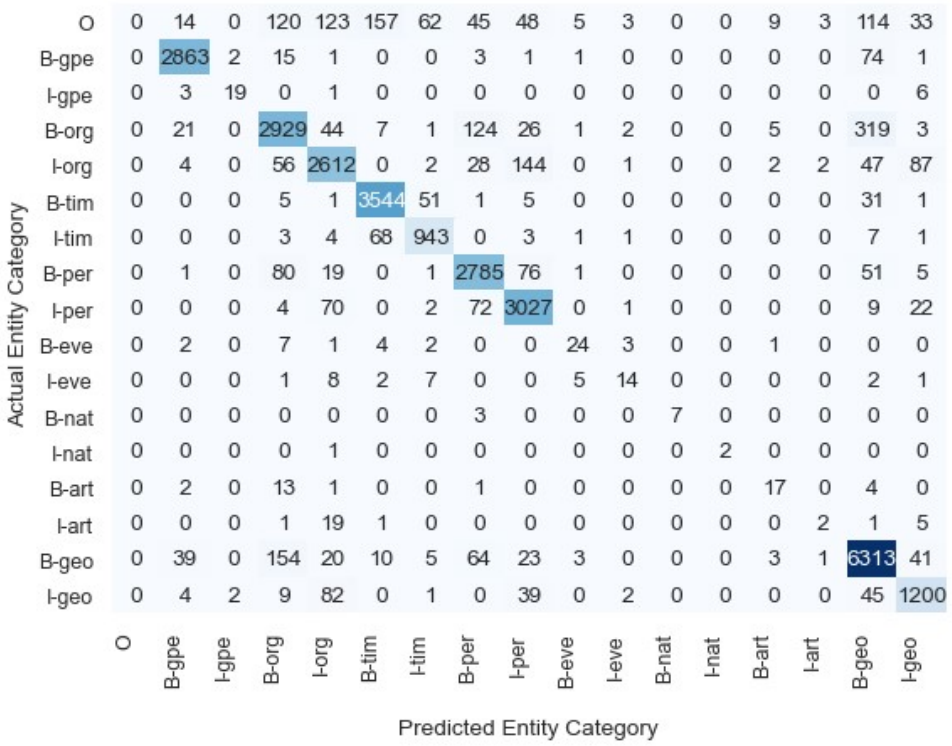


Fig. 5.6. Confusion matrix of BERT model for the baseline approach (model confidence only)

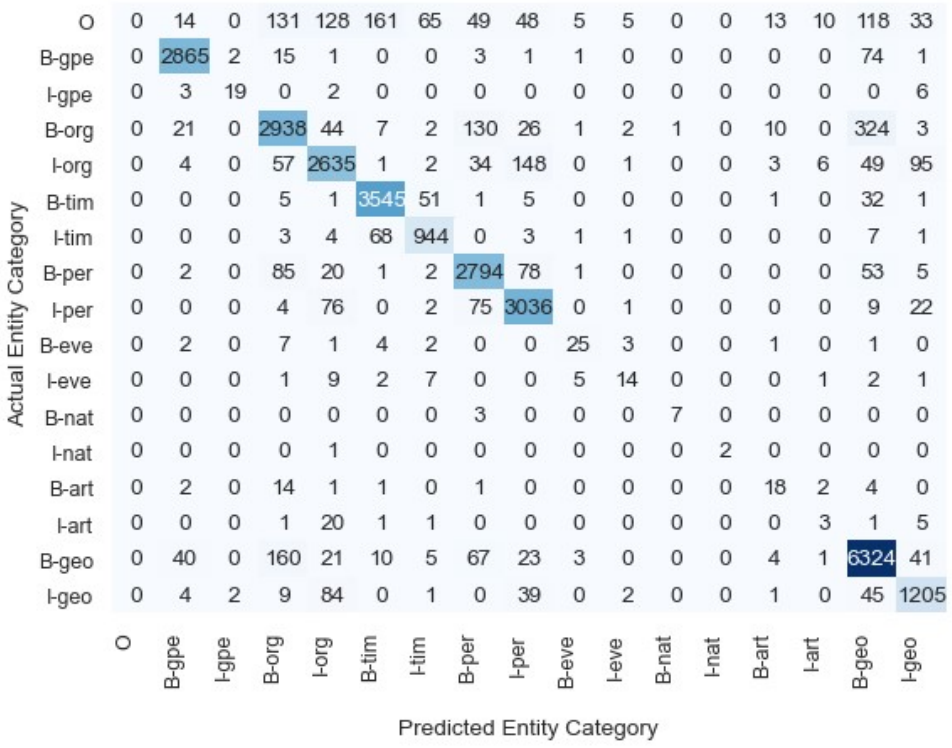


Fig. 5.7. Confusion matrix of BERT model for the proposed approach (model confidence + data confidence)

Meanwhile, we also see the same trend as that in the WNUT'17 dataset; with more entities receiving high prediction scores, extra errors incurred. But the models still demonstrated very competitive prediction accuracy. One difference between the WNUT'17 result and the GMB result was that the increasing of the entities receiving high prediction confidence in GMB was not as many as that in WNUT'17. This should be due to the different dataset characteristics as aforementioned and the BERT model adopted which is one of the best deep learning models today for text data processing and therefore, harder to further improve.

Besides the results presented in this chapter, we also conducted experiments based on different *alpha* and *threshold* values. Similar to those shown in Table 5.4 and 5.3 and Figs. 5.4 to 5.7, the proposed model performed the best in most cases when *alpha* = 0.95 and *threshold* = 0.85. Thus, we skip other results in this thesis.

# CHAPTER 6

# CONCLUSION

This thesis originated from a particular application field of machine learning — the recommendation systems. Sponsored by Intuit Inc., we targeted a relatively understudied problem of estimating the reliability of modern deep learning model predictions. We intended to draw attention to this particular area of machine learning because of the recent advancements in machine learning approaches, especially deep neural networks, can misled users about the practical applicability of these new machine learning tools. Considering the Intuit's research problem and practical application needs, we proposed a novel approach to significantly enhance the learning model's prediction confidence score. The approach efficiently combined the traditional Softmax scores generated by deep neural networks with data statistics based confidence estimation. To evaluate the proposed method, we conducted a case study on a popular natural language processing problem—Named Entity Recognition. We targeted this problem because it is of the particular interest of Intuit Inc. and many industry companies who need to know the reliability of their trained models before releasing them to the customers. We also designed a new evaluation metric TEC to measure the model performances on prediction confidence estimation. Using a variety of state-of-the-art deep learning models, we applied the proposed approach to two popular NER datasets: WNUT'17 and GMB. Experimental results showed that our approach can effectively improve the number of instances receiving high prediction confidence while maintaining highly competitive model

performance in terms of accuracy. Thus, it can drastically reduce the workload and cost for manual examination of the instances with low prediction scores. We believe that this work will inspire the machine learning community to pay more attention to this problem and thereafter advance the field by proposing new approaches to estimate model prediction reliability.

# REFERENCES

[1] D. Sarkar, “Robust Word2Vec Models with Gensim & Applying Word2Vec Features for Machine Learning Tasks.” https://www.kdnuggets.com/2018/04/ robust-word2vec-models-gensim.html, 2018. [Online; accessed 9-November2022].

[2] D. Sarkar, “Implementing Deep Learning Methods and Feature Engineering for Text Data: The GloVe Model.” https://www.kdnuggets.com/2018/04/ implementing-deep-learning-methods-feature-engineering-text-data-glove. html, 2018. [Online; accessed 9-November-2022].

[3] M. Woolf, “Pretrained Character Embeddings for Deep Learning and Automatic Text Generation.” https://minimaxir.com/2017/04/char-embeddings/, 2017. [Online; accessed 9-November-2022].

[4] “Deep Convolutional Neural Networks - A Guide.” https: //www.run.ai/guides/deep-learning-for-computer-vision/ deep-convolutional-neural-networks, 2018. [Online; accessed 16November-2022].

[5] R. Hachilif, R. Baghdadi, and F. Benhamida, *Graduation Thesis Implementing and Optimizing Neural Networks using Tiramisu*. PhD thesis, 06 2019.

[6] D. Liu, “A Practical Guide to ReLU.” https://medium.com/@danqing/ a-practical-guide-to-relu-b83ca804f1f7, 2017. [Online; accessed 9November-2022].

[7] “Max-pooling / Pooling.” https://computersciencewiki.org/index.php/ Max-pooling_/_Pooling, 2018. [Online; accessed 9-November-2022].

[8] D. Radeˇci´c, “Softmax Activation Function Explained.” https://towardsdatascience.com/

softmax-activation-function-explained-a7e1bc3ad60, 2020. [Online; accessed 9-November-2022].

[9] X. Ma and E. Hovy, “End-to-end sequence labeling via bi-directional LSTMCNNs-CRF,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 1064–1074, Association for Computational Linguistics, Aug. 2016.

[10] J. Alammar, “The Illustrated Transformer.” https://jalammar.github.io/ illustrated-transformer/, 2018. [Online; accessed 9-November-2022].

[11] “Evaluation measures for multiclass problems.” http://gabrielelanaro. github.io/blog/2016/02/03/multiclass-evaluation-measures.html, 2016. [Online; accessed 9-November-2022].

[12] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[13] C. A. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[14] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, *et al.*, “The youtube video recommendation system,” in *Proceedings of the fourth ACM conference on Recommender systems*, pp. 293–296, 2010.

[15] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, “Wide & deep learning for recommender

systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

[16] C. Musto, C. Greco, A. Suglia, and G. Semeraro, "Ask me any rating: A contentbased recommender system based on recurrent neural networks.," in *IIR*, 2016.

[17] A. Suglia, C. Greco, C. Musto, M. De Gemmis, P. Lops, and G. Semeraro, "A deep architecture for content-based recommendations exploiting recurrent neural networks," in *Proceedings of the 25th conference on user modeling, adaptation and personalization*, pp. 202–211, 2017.

[18] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," *Advances in neural information processing systems*, vol. 26, 2013.

[19] G. K. Dziugaite and D. M. Roy, "Neural network matrix factorization," *arXiv preprint arXiv:1511.06443*, 2015.

[20] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, pp. 111–112, 2015.

[21] Y. Ouyang, W. Liu, W. Rong, and Z. Xiong, "Autoencoder-based collaborative filtering," in *International conference on neural information processing*, pp. 284–291, Springer, 2014.

[22] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2051–2059, 2017.

[23] S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, and B. Xu, “Joint extraction of entities and relations based on a novel tagging scheme,” *arXiv preprint arXiv:1706.05075*, 2017.

[24] S. Seo, J. Huang, H. Yang, and Y. Liu, “Interpretable convolutional neural networks with dual local and global attention for review rating prediction,” in *Proceedings of the eleventh ACM conference on recommender systems*, pp. 297– 305, 2017.

[25] S. Okura, Y. Tagami, S. Ono, and A. Tajima, “Embedding-based news recommendation for millions of users,” in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1933–1942, 2017.

[26] J. Tan, X. Wan, and J. Xiao, “A neural network approach to quote recommendation in writings,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 65–74, 2016.

[27] C.-Y. Wu, A. Ahmed, A. Beutel, and A. J. Smola, “Joint training of ratings and reviews with recurrent recommender networks,” 2017.

[28] F. Ortega, R. Lara-Cabrera, A. Gonza´lez-Prieto, and J. Bobadilla, “Provid-´ ing reliability in recommender systems through bernoulli matrix factorization,” *Information Sciences*, vol. 553, pp. 110–128, 2021.

[29] J. Bobadilla, A. Guti´errez, F. Ortega, and B. Zhu, “Reliability quality measures for recommender systems,” *Information Sciences*, vol. 442, pp. 145–157, 2018.

[30] A. Hernando, J. Bobadilla, F. Ortega, and J. Tejedor, “Incorporating reliability measurements into the predictions of a recommender system,” *Information Sciences*, vol. 218, pp. 1–16, 2013.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, highperformance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[34] M. Munikar, S. Shakya, and A. Shrestha, "Fine-grained sentiment classification using bert," in *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, vol. 1, pp. 1–5, IEEE, 2019.

[35] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

[36] C. Chen, X. Meng, Z. Xu, and T. Lukasiewicz, "Location-aware personalized news recommendation with deep semantic analysis," *IEEE Access*, vol. 5, pp. 1624–1638, 2017.

[37] W. Niu, J. Caverlee, and H. Lu, "Neural personalized ranking for image recommendation," in *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 423–431, 2018.

[38] T. Alashkar, S. Jiang, S. Wang, and Y. Fu, "Examples-rules guided deep neural network for makeup recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[39] D. Liang, M. Zhan, and D. P. Ellis, “Content-aware collaborative music recommendation using pre-trained neural networks.,” in *ISMIR*, pp. 295–301, 2015.

[40] W. Huang, Z. Wu, C. Liang, P. Mitra, and C. L. Giles, “A neural probabilistic model for context based citation recommendation,” in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[41] X. Wang, X. He, L. Nie, and T.-S. Chua, “Item silk road: Recommending items from information domains to social users,” in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 185–194, 2017.

[42] T. Ebesu and Y. Fang, “Neural citation network for context-aware citation recommendation,” in *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pp. 1093–1096, 2017.

[43] Q. Liu, S. Wu, and L. Wang, “Deepstyle: Learning user preferences for visual recommendation,” in *Proceedings of the 40th international acm sigir conference on research and development in information retrieval*, pp. 841–844, 2017.

[44] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, “What your images reveal: Exploiting visual contents for point-of-interest recommendation,” in *Proceedings of the 26th international conference on world wide web*, pp. 391– 400, 2017.

[45] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, “Image-based recommendations on styles and substitutes,” in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

[46] J. Wen, X. Li, J. She, S. Park, and M. Cheung, “Visual background recommendation for dance performances using dancer-shared images,” in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 521–527, IEEE, 2016.

[47] J. Zhou, C. Gurrin, and R. Albatal, “Applying visual user interest pro les for recommendation & personalisation.(2016),” 2016.

[48] R. He and J. McAuley, “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering,” in *proceedings of the 25th international conference on world wide web*, pp. 507–517, 2016.

[49] R. He and J. McAuley, “Vbpr: visual bayesian personalized ranking from implicit feedback,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[50] W.-T. Chu and Y.-L. Tsai, “A hybrid recommendation system considering visual information for predicting favorite restaurants,” *World Wide Web*, vol. 20, no. 6, pp. 1313–1331, 2017.

[51] W. Yu, H. Zhang, X. He, X. Chen, L. Xiong, and Z. Qin, “Aesthetic-based clothing recommendation,” in *Proceedings of the 2018 world wide web conference*, pp. 649–658, 2018.

[52] J. Lee, S. Abu-El-Haija, B. Varadarajan, and A. Natsev, “Collaborative deep metric learning for video understanding,” in *Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining*, pp. 481–490, 2018.

[53] S. Cao, N. Yang, and Z. Liu, “Online news recommender based on stacked autoencoder,” in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pp. 721–726, IEEE, 2017.

[54] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, “On deep learning for trust-aware recommendations in social networks,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1164–1177, 2016.

[55] Y. Pan, F. He, and H. Yu, “Trust-aware collaborative denoising auto-encoder for top-n recommendation,” *arXiv preprint arXiv:1703.01760*, 2017.

[56] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, “Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention,” in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 335–344, 2017.

[57] Y. Tay, A. T. Luu, and S. C. Hui, “Multi-pointer co-attention networks for recommendation,” in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2309–2318, 2018.

[58] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, *et al.*, “A survey of uncertainty in deep neural networks,” *arXiv preprint arXiv:2107.03342*, 2021.

[59] M. Sensoy, L. Kaplan, and M. Kandemir, “Evidential deep learning to quantify classification uncertainty,” *Advances in neural information processing systems*, vol. 31, 2018.

[60] A. Malinin and M. Gales, “Predictive uncertainty estimation via prior networks,” *Advances in neural information processing systems*, vol. 31, 2018.

[61] M. Raghu, K. Blumer, R. Sayres, Z. Obermeyer, B. Kleinberg, S. Mullainathan, and J. Kleinberg, “Direct uncertainty prediction for medical second opinions,” in *International Conference on Machine Learning*, pp. 5281–5290, PMLR, 2019.

[62] T. Ramalho and M. Miranda, “Density estimation in representation space to predict model uncertainty,” in *International Workshop on Engineering Dependable and Secure Machine Learning Systems*, pp. 84–96, Springer, 2020.

[63] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, “A simple baseline for bayesian uncertainty in deep learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[64] K. Osawa, S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota, “Practical deep learning with bayesian principles,” *Advances in neural information processing systems*, vol. 32, 2019.

[65] M. Leutbecher and T. N. Palmer, “Ensemble forecasting,” *Journal of computational physics*, vol. 227, no. 7, pp. 3515–3539, 2008.

[66] W. S. Parker, “Ensemble modeling, uncertainty and robust predictions,” *Wiley Interdisciplinary Reviews: Climate Change*, vol. 4, no. 3, pp. 213–223, 2013.

[67] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, “Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation,” in *International MICCAI Brainlesion Workshop*, pp. 61–72, Springer, 2018.

[68] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren,“Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks,” *Neurocomputing*, vol. 338, pp. 34–45, 2019.

[69] L. Zhang and Y. Piao, “Crf-based figure/ground segmentation with pixel-level sparse coding and neighborhood interactions,” *Journal of information and communication convergence engineering*, vol. 13, no. 3, pp. 205–214, 2015.

[70] N. Plath, M. Toussaint, and S. Nakajima, “Multi-class image segmentation using conditional random fields and global classification,” in *Proceedings of the 26th annual international conference on machine learning*, pp. 817–824, 2009.

[71] T. Liu, X. Huang, and J. Ma, “Conditional random fields for image labeling,” *Mathematical Problems in Engineering*, vol. 2016, 2016.

[72] Z. Yun-tao, G. Ling, and W. Yong-cheng, “An improved tf-idf approach for text classification,” *Journal of Zhejiang University-Science A*, vol. 6, no. 1, pp. 49– 55, 2005.

[73] F. M. Couto, M. J. Silva, and P. M. Coutinho, “Finding genomic ontology terms in text using evidence content,” *BMC bioinformatics*, vol. 6, no. 1, pp. 1–6, 2005.

[74] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *International conference on machine learning*, pp. 1321– 1330, PMLR, 2017.

[75] X. Ma and E. Hovy, “End-to-end sequence labeling via bi-directional lstm-cnnscrf,” *arXiv preprint arXiv:1603.01354*, 2016.

[76] A. Graves, S. Fern´andez, and J. Schmidhuber, “Bidirectional lstm networks for improved phoneme classification and recognition,” in *International conference on artificial neural networks*, pp. 799–804, Springer, 2005.

[77] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, “Results of the wnut2017 shared task on novel and emerging entity recognition,” in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 140–147, 2017.

[78] J. Bos, V. Basile, K. Evang, N. J. Venhuizen, and J. Bjerva, “The groningen meaning bank,” in *Handbook of linguistic annotation*, pp. 463–496, Springer, 2017.

[79] “pytorch-NER.” https://github.com/cswangjiawei/pytorch-NER?fbclid=IwAR3r4YMbVrv1TWaRPtqscJI3HyEXwUV1vhoAARYW0Co2YRDD9l_rng35EOs, 2020. [Online; accessed 9-November-2022].

[80] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, “Huggingface’s transformers: State-ofthe-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.

# CURRICULUM VITA

## Prianka Banik

Prairie View A&M University
Department of Computer Science
P.O. Box 519, Prairie View, 77446
Phone: (832) 416-9488
Email: prianka.banik.buet@gmail.com

## Education

**Prairie View A&M University**, USA
*Masters of Science in Computer Science · (Present, expected graduation date Dec'22)*

**Bangladesh University of Engineering and Technology**, Bangladesh *Bachelor of Science in Computer Science and Engineering · (2013)*

## Experience

**Software Engineer, Mobile Lab 2,Samsung R&D Institute of Bangladesh, Dhaka (August 1, 2013 - August 31, 2015)**

## Technical Skills

**Languages :** Python, C++, Java

**Database :** MySQL

**Deep Learning Frameworks :** Pytorch

## Publications

**Banik P**, Dong X, Qian L, Li L, Enhancing Prediction Reliability of Deep Learning by Data Confidence for Recommendation Systems: A Case Study on Named Entity Recognition, *Under Preparation*, 2022.

Naha S, Xiao Q and **Banik P** and Reza M A, Crandall, D J. Part segmentation of novel quadruped animals using keypoint guidance. In IEEE Winter Conference on Applications of Computer Vision (WACV), 2021.

Rahman S, **Banik P**, Naha S. LDA based paper currency recognition system using edge histogram descriptor. In 2014 17th International Conference on Computer and Information Technology (ICCIT) 2014 Dec 22 (pp. 326-331). IEEE.