Multitasking Learning With Unreliable Labels

Debjit Paul

Department of Computer Science

Saarland University

Masters Thesis

Supervisor:
Prof.Dr.Dietrich Klakow

Reviewers:
Prof.Dr.Dietrich Klakow
Prof.Dr.Klaus Berberich

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,_____ _____

(Date/Datum) (Unterschrift/Signature)

Acknowledgements

To begin with, I would like to take this opportunity to thank my parents for their unconditional love and untiring support.

I would like to thank my advisor and supervisor Prof.Dr. Dietrich Klakow, for making this thesis a fulfilling learning opportunity for me. I am grateful towards him for his excellent supervision during the course of the thesis helping me with his constant guidance and constructive criticisms.

I am highly grateful to Prof.Dr.Klaus Berberich for reviewing this thesis and also for the intuitions and knowledge shared during the intriguing discussions.

A special thanks to my friend and colleague Mittul Singh for his support and his ideas during the experiments, scientific writing and also to proof read my thesis document.

My dear friends - Manjima Bardhan, Russa Biswas, Sreyasi Nag Chowdhury, Rohit Kumar Gupta, and Soumya Chakraborty- had gone out of there way to help me in improving the manuscript. Thank you all for the support and valuable suggestions.

Lastly, I would like to extend my deepest gratitude to my family and close friends for their support during the difficult times of my thesis.

Abstract

Neural networks have shown impressive performance for various classification tasks due to the availability of large datasets. However, a known problem in classification tasks is the presence of unreliable labels. Successive manual annotations can improve the quality of the data set but are expensive to obtain. Hence, most applications rely on the artificial annotation (e.g., Brill tagger for POS tags) of large data. However, the labels generated by the artificial tagger are unreliable or noisy. Previously Noisy Label Neural Network (NLNN) algorithm has been used to alleviate this issue on a single classification task. We can further reduce this noise by also including supervision signals for related classification tasks in this setup. To this aim, we explore extending the single-task based noise-reduction setup to a multi-task case. Hence, we build a Multi-Task Learning-based (MTL) Noisy Label Neural Network. In this work, we establish Chunking as the primary task with POS tag classification as the auxiliary task. We set up the data by adding artificially annotated data with the human annotated Penn Treebank data. In our experiments, we compare the MTL-based setup with and without noise reduction against NLNN on the single task. We show that MTL-based NLNN outperforms the state-of-the-art for Chunking.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, the increase in the size of the data in the world has become overwhelming to the scientist. The massive growth in the amount of data that can be accessed due to the World Wide Web has partially fulfilled the dream of decreasing the gap made between computers and humans. Figure 1.1, depicts the relationship between the increase in size of the data and deep neural net (advanced machine learning approach). The advancement in the field of machine learning has helped to solve several tasks such as face-recognition and object-recognition, not only in photographs but also in live videos, detection of latent topics in natural language texts, fraud detection, online search and online recommendation system. The enormous increase in the computational power has made the task significantly faster. Applications of neural net has become common due to the increase in usage of GPUs, and most recently Google's TPU (Tensor Processing Unit). Hence, the increase in the magnitude of data, computational power and improved machine learning algorithms has a synergistic effect on the breakthrough that is made by the field of machine learning in various areas.

Machine Learning(ML) emphasizes on automatic processing from general observation. Alike human learning, machine learning algorithms can improve their algorithm by themselves with the help of information and knowledge. In the last decade, there has been a shift from knowledge-driven approach to data-driven approach in ML. There has been much progress made in solving many complex tasks using machine learning. One of the important success stories of machine learning is the breaking through Pattern Recognition and Natural language Processing (NLP). NLP bridges the gap between human speech and computer understanding. The amalgamation of ML and NLP assists the computer not only to understand the complicated human language but also to efficiently hold a conversation. There has been a lot of research work put into natural language understanding. To represent the natural language

meaning to the computer, there are several NLP (Part-of-speech, name entity, semantics) and ML (classification) techniques that are used extensively.

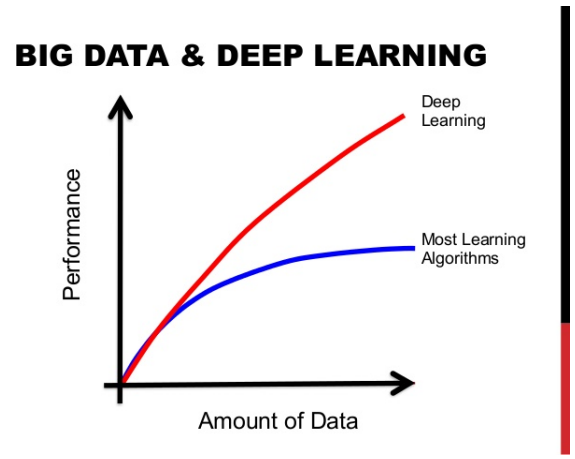

Figure 1.1: Big Data vs Deep Learning [6]

Classification technique is one of the paradigms of Machine Learning which identifies the concept for new instances with the help of fixed given instances (observations) related to it. Figure 1.2, depicts the work-flow of supervised classification. The observation plays a key role in classification task. It is important to understand the usefulness of labeled training data which is an input along with the features to the machine learning algorithms, to learn and predict labels for test data.

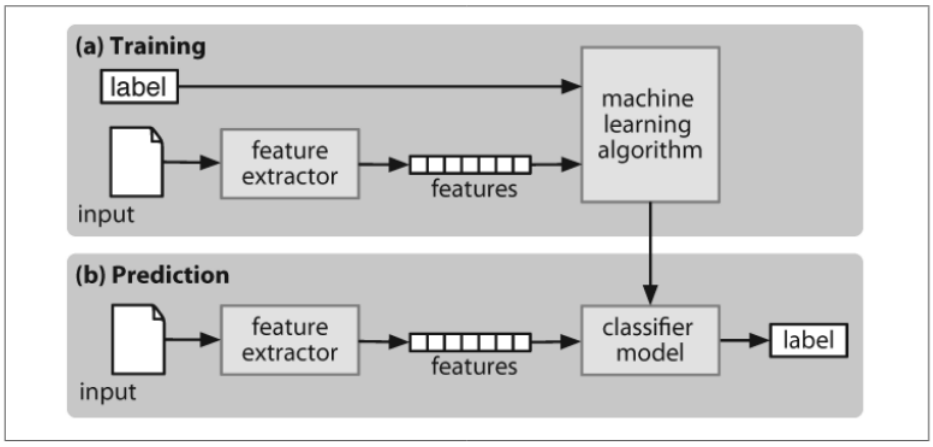

Figure 1.2: Work flow of supervised classification [39]

One of the most fundamental and yet an important NLP problem is Part-of-speech tagging. Part-of-speech carries a lot of information about a word and hence, plays a crucial role in providing natural language meaning to the computer. For many decades, there has been a lot of research dedicated to building a system which will encode POS [27]. In languages like English and German, various tagged corpuses are present. Although a lot of manual work went into tagging the existing corpora, currently deep learning methods are used for automatic tagging. The state-of-the-art tagging performance for popular language like English and German are significantly high when using such Deep Learning approaches(approximately 97 percentage) [27].

Tagging problems like chunking, POS tagging are treated as a sequence labeling problem or a classification problem [27]. Lately, a term has been coined known as 'Multi-tasking Learning' (MTL) which involves training two or more tasks simultaneously over shared representation [42]. The motivation is applying our learning from auxiliary tasks to the main task. Moreover, with joint learning a model can perform classification task for two or more task simultaneously. However, as we mentioned above, classification problems need observations. These observations can be manually (human annotated) or artificially generated. The human annotated labels or observations are reliable and correct, nevertheless, for a large data it is very expensive and time-consuming. Comparatively, the artificial annotation labels are less expensive. However, artificial annotation introduces noisy labels which are assumed to be reliable during training. Therefore, they might be misleading information that will subvert the model.

In [47] X.Zhu & X.Wu. has shown that the noisy labels are more harmful than noisy features. Bekker & Goldberger [7] proposed one solution to this problem, known as the Noisy Labels Neural Network (NLNN) which estimates the noise distribution in the noisy labels and neural network parameters in the training phase. Bekker & Goldberger evaluated NLNN on MNIST handwriting recognition dataset and TIMIT phoneme classification dataset by injecting label noise. They injected noise in labels by converting clean labels into noisy labels using stochastic methods. However, this thesis aims to use and explore the effectiveness of NLNN on noise generated by artificial taggers for Part-of-speech tagging and chunking. Such approach can be unitized to increase labelled data for a low resource language dataset.

Multi-tasking Learning is a machine learning approach which thrives on the supervised signals produced by other related tasks to improve generalization performance

[32]. Multi-tasking Learning is used for several NLP tasks to improve the performance and get a good representation. In order to handle noisy labels we explore MTL for the supervision signals from the related task. To make the system more noise-robust, we setup an system to use both MTL and NLNN, such that NLNN can be used to generate clean labels for two or more tasks simultaneously.

In [7] showed NLNN is robust to noise labels with unknown distribution for image dataset and speech dataset. The application of NLNN can arouse the following research questions:

- Does NLNN produce impressive results on a complex linguistic task?
- Does NLNN produce impressive results on noise in artificial labels?
- Can NLNN be effective in handling noisy label for multiple task simultaneously?

In order to answer the above research questions, we performed the following experiments:

- An application of the NLNN method on POS tagging fixing the training data and changing size of the artificial labelled data.
- An application of the NLNN method on POS tagging when small amount of clean labelled data is added and when large amount of clean labelled data is added.
- An application of the NLNN method on Chunking.
- An application of the NLNN method on Higher level task Chunking with low supervision of low-level task (Multi-tasking learning).

We performed the first experiment to address the robustness of NLNN for POS tagging, when the artificial labelled data replace the clean labelled data. In most of the alternative work proposed earlier to handle noise labels, they added noise labels by corrupting the clean labels. In this experiment we inject noise by replacing the clean labels with artificial labelled data.

The next step is to address the real-life problem, where we have low-resource data. We fix the amount of clean labeled data, and we add artificial labeled data on top of it. In this experiment, we address the problem of how much artificial labeled data

can be cleaned such that it can be re-used as cleaner data. We changed the size of the clean labeled data from 100k to 1 billion words and proportionally the amount of artificial labeled data, to see the robustness of NLNN for POS Tagging.

In the next experiment, we addresses the above problem for a simpler task (Chunking). The main goal is to check the variance in performance of NLNN when move to a simpler task.

As we mentioned earlier, NLNN was proposed for Single task learning method. In this thesis, we extended NLNN to Multi-task learning based method where we apply it for POS tagging and chunking problem. We explored the robustness of NLNN for two separate noise labelled dataset.

Before elaborating the experiments in section details, in this thesis chapter 2 presents the general background of research in this direction and overview on MTL, followed by a brief introduction to noisy label neural networks with EM algorithm in chapter 3. We elaborate on Multi-tasking learning in Sequence Learning in Chapter 4. Chapter 5 contain detailed descriptions of experiments and results. The thesis ends with a constructive discussion about this work and possible future research directions.

Chapter 2

Background

In the last decade, the use of machine learning has increased drastically throughout computer science and beyond. Machine learning automatically learns programs from observations (training data). The inference of human being in machine learning is intended to provide training data. Preferably, with as little involvement as possible. The idea is to provide enough information (training data) to learn such that it can be used to categorize test data.

Machine Learning (ML) models try to infer knowledge from the training data. When the ML algorithms are trained on datasets, it is essential to provide a relevant set of data rather than giving a massive size of data. With the advancement of machine learning models, automatic understanding of data and extracting knowledge has become an important feature. In ML, for supervised and semi-supervised approaches features and labels are fully or partially provided. These approaches learn a mapping function from features to labels.

In natural language processing, the labels are often known as annotations. These annotations are meta-data which provides information about the text in the corpora. The computers are useful in finding patterns in data, with the help of meta-data. Although, the web contains many text data, correctly annotated data are rare. There has been much linguistic research work going around to relive the deeper meaning of a text and find more metadata tag (annotations) in the corpus. However, due to the increase in the size of data, this problem is viewed as ML modeling problem. In NLP, datasets are commonly known as corpora and annotated dataset is referred to annotated corpus. Machine Learning algorithms can be trained on such annotated corpora.

The linguistic features such as part-of-speech, chunking, name-entity tagging, etc. are rich enough to capture the behavior of words and phrases. These features are used as annotation values of any language. Stubbs, Amber C. in [36] proposed a method named as MATTER methodology, for creating annotations using machine learning task. There are cycle of development which are follows:

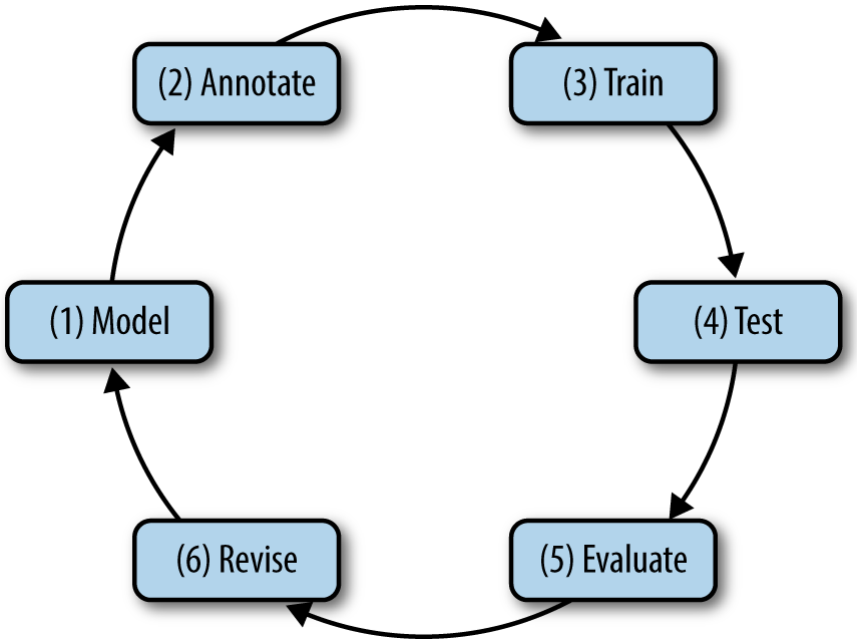

Figure 2.1: MATTER methodology

- **Model** It represents the information collected from the corpus to perform annotations.
- **Annotate** It represents the guidelines to annotate, finding and searching annotations.
- **Train** It represents the training phase of a ML model on a corpus with some features.
- **Test** It represents the testing the strength of the model after the training.
- **Evaluate** It represents the accuracy of the model on the test set.
- **Revise** It represents the revision the entire projects from ML model to the corpus.

There are many advanced unsupervised approaches which don't require labeled data (annotations). EM algorithm and various clustering (k-mean, spectral clustering, hierarchical, etc.) techniques are the popular unsupervised approaches. The concept behind clustering is to form groups of data points which are similar among themselves in some respect. The performance of unsupervised approaches is impressive for unlabeled data. However, in our work to learn from un-reliable annotations, we need to explore supervised and unsupervised approach. In such setup, the supervised method is used to learn a good representation of the data, and unsupervised method is used to clean the noise in the data.

In supervised approaches often we assume that the labels are accurate and unambiguous [17]. However, for real life problem, it is not the case always as performing annotations on a corpus is an expensive and time-consuming task [36]. Performing annotations on domain-based corpus such as medical training data become even more difficult. In practice, even the human annotators due to lack of knowledge and time make mistakes, and then the machine learning algorithm still assumes them as "gold standards." Noise labeled data is more harmful compare to noise features and missing data [47]. Machine Learning algorithms find handling noise labeled training data most difficult to learn [13]. In our work, we intend to develop and explore noise-robust machine learning techniques. In the next section, we will explore about noise labels in details.

2.1 Noisy Label

Training data is an integral part of any machine learning algorithm. In supervised learning, the training data contains pairs of input objects (features) and an output value(annotations, labels). Due to lack of time and knowledge often researchers use artificial annotators to annotate the input vectors. The artificial annotators introduce unreliable or noise labels in the training data. While training a machine learning classifier on input data with unreliable labels, it fails to capture the underlying mapping function between the features and labels. Hence, the classifiers fail to produce the desired predictions [13]. Figure 2.3, depicting an example of noise labels for POS tagging. As we can see there are two kinds of labels one is true and another is noisy. In this example the word 'is' is wrongly tagged as 'VBG'(Verb, gerund or present

participle) depicting the noise label (mislabeling) but the true label would be 'VBZ' (Verb, 3rd person singular present).

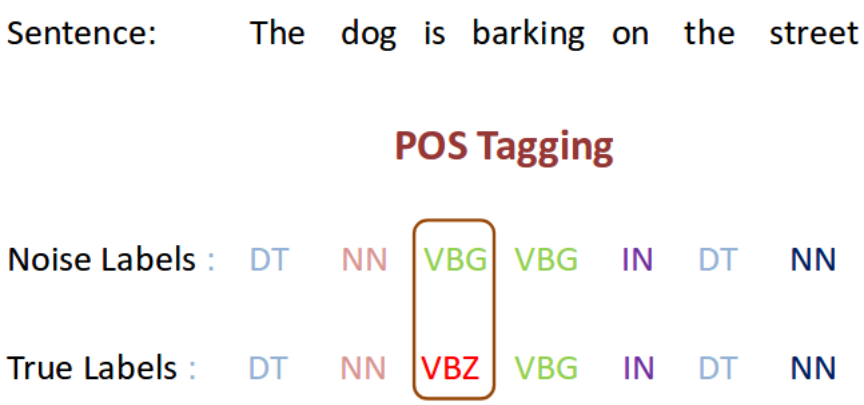

Figure 2.2: Example of Noise Labels for POS Tagging

2.2 Types of Noisy Labels

There are various ways to look at the problem of noisy labels, which are as following:

1. **Inconsistency Labelling**

   Label inconsistency may occur due to the difference in predicted output and expected output given the input features. Some input features may have multiple labels or there are complex structures in training data which are responsible for such discrepancy. Figure 2.4 shows an example of inconsistency labelling.

2. **Mislabelling**

   It addresses to the dataset containing corrupted labels. Even the best of ML classifiers performance deteriorates due to inherent class label noise [13]. In several bio-medical dataset mislabelling has been reported. Due to the presence of low resource of dataset mislabelling can significantly impact the ML classifier.

3. **Outlier**

   An outlier are normally stated as the data point which maintains an abnormal distance from the other data points. Noisy Labels can be addressed as outlier if

an instance which has unusual characteristics and is tagged with a wrong label. Sukhbaatar and Fergusk [40] proposed a method to counter outlier noise labels.

4. **Missing Labels**

   Missing Labels can also be viewed as noise labels. There are various cases where the input features are not annotated properly. It can be solved by semi-supervised approach [44].

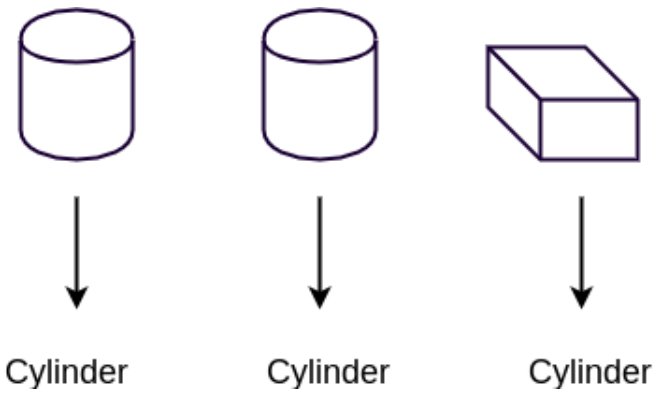

Figure 2.3: Example of Inconsistency Labelling

The term **noise labels** often known as mislabeling, in other words, a input vector is wrongly tagged. There are common assumption on noise labels [40] which are as follows:

- Noise in labels are entirely random (simple),
- Noise in labels random given the true label (average),
- Noise in labels depends on actual input itself (complex problem)

Figure 2.2, portray the dependency graph of noise labels. X is the input vector, Y is the true labels and Z is the noise labels in figure 2.2 [40]. Z can either independent or depend on Y and X. In our work, we explored NLNN where noise labels are only depends on Y.

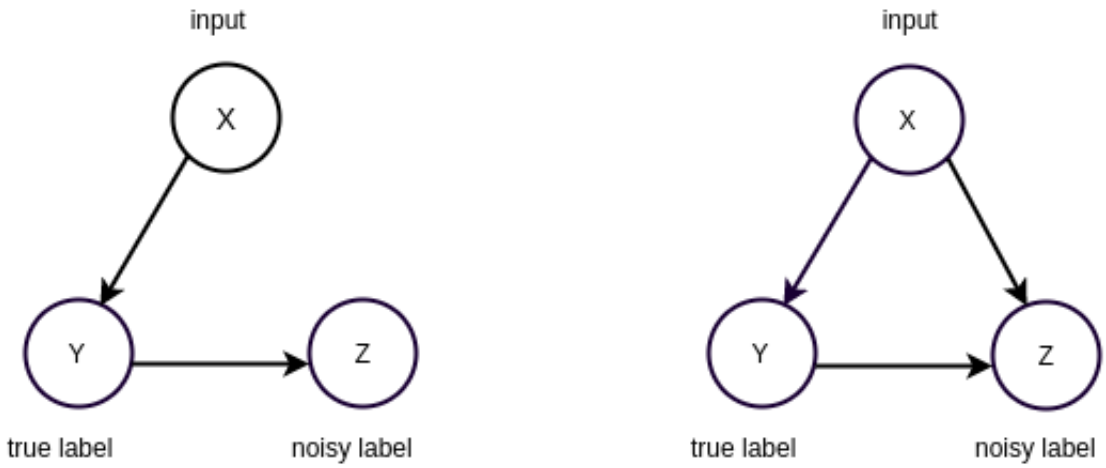

Figure 2.4: Dependencies of Noisy Labels

2.3 Artificially Generating Noise Labels

Uniform Noise

Random noise are often modeled as uniform noise. Uniform noise takes an error-rate as input which depicts the amount of noise present in the training data. Then this error-rate is distributed to each class uniformly. Therefore, each class labels has equal likelihood of finding noise label. Random noise labels are assumed to either independent of the input features and the correct labels or dependent on only the correct labels. However, uniform noise fails to address the situation when noise labels are dependent on correct labels. .

Permutation Noise

On the other hand, permutation Noise can model the noise which depends on the correct label. Permutation noise use a probability p to transform each correct label to a different label. However, the permutation noise doesn't model the noise label when it depends on both input features and true labels. Modelling noise depending on the input features are complex problem to solve.

Artificial Annotators

Most of the artificial annotators can produce noise depending on the input features. Depending on the complexity of task and complex structures of training data, artificial annotators do produce noise labels. Artificial annotators can be Brill Tagger for POS tagging. In this thesis, we produced noise labels using artificial annotators and

explored the robustness of NLNN on those noise labels.

In general, it would be desirable to model human annotators. Another preferable idea will be generating noise dataset with a prior knowledge of the real-world noise, such that any noise-robust model can be tested on it. A generative model was proposed by Hovy et al. (MACE)[16] to determine which annotators to trust. MACE (Multi-Annotator Competence Estimation)is graphical model.

2.4 Noise-robust techniques

In recent times, there has been lot of approaches proposed to make deep learning robust to noise in labels. There has been different approaches to predict noisy labels, one of them was by Larsen et al [20]. They assumed the generation of noise is independent of class labels and the features. They proposed noise as outlier probability and learned using probabilistic modelling and back propagation.

Mnih & Hinton [25] proposed an advanced noise-robust model which uses robust loss functions. This method can handle missing class problems. A similar approach was proposed by Lee [21] using minimum entropy regularization. They used pseudo-labels as training data for unlabeled data. However, in both the approaches they considered binary classification. Reed et al. [33] introduced a simple approach using bootstrapping scheme to handle noise and incomplete labels. In this work they labeled the unlabeled based on a small list of seeds (labels).

In [44], Grandvalent & Bengio addressed missing labels problem which can seen as label noise problem. They developed a semi-supervised algorithm that could predict the non-labeled data. Natarajan [26] suggested a unbaised estimator to learn with noisy labels for binary classification problem.

A similar approach to NLNN [7] was proposed by Sukhbaatar and Fergusk [40] to model noise with linear layer. They introduce an extra linear layer whose weight matrix has the shape of a noise distribution. However, the additional linear layer does not constitute a model of a noisy channel. The extra layer tries to match the noise label distribution. In there experiments [40], the performed image classification task

using Convolutional Neural Network (CNN). They introduced noise in labels by:

- Flipping Changing the label into another label within the set of labels. For example, label set =cat,dog,horse, Changing the tag for Horse image to cat, Cat image to dog, etc.
- Outliers Tagging the image which doen't belong to the set of labels is tag with one of the labels. For example, label set =cat,dog,horse, An outlier image of Flower tagged as cat.

Sukhbaatar and Fergusk [40] proposed two method to handle noise which are as follows:

- Bottom up Noise Model

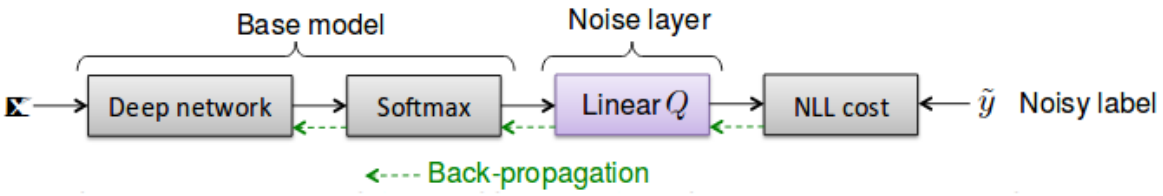

Figure 2.5: Sukhbaatar & Fergus bottom_up noise-robust NN [40]

In the bottom up noise model, noise layer was placed after the softmax layer. Hence, noise distribution can update with the network's output. The noise layer is a implemented by a linear layer.

- Top Down Noise Model

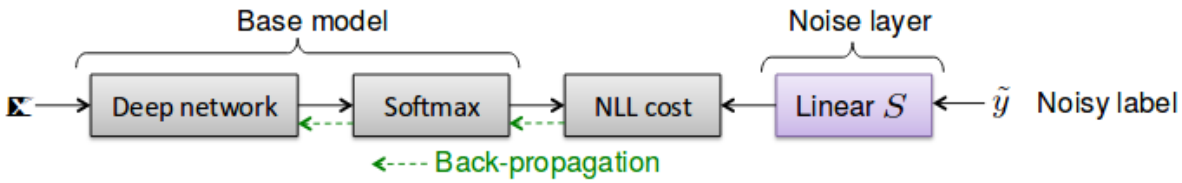

Figure 2.6: Sukhbaatar & Fergus Top-down noise-robust NN [40]

In the top-down noise model, noisy labels is placed after the cost layer. It couldn't learn linear layer using standard backpropagation. Noise labels goes through a conversion to influence the training method.

Recently, Jacob Goldberger and Ehud Ben-Reuven proposed a new version of NLNN (Noisy Label Neural Network). In this work, they introduced a noise adaptation layer. This model assumes the noisy labels are depending on both correct labels and the input features. They introduced noise into labels by stochastically changing some clean labels to noise labels.

In this work, they introduced an alternative to EM algorithm which was used in their previous work. In the training phase, they introduced a second softmax layer which can be seen as a noise adaptation layer to provide an alternation to EM. NLNN with EM algorithm used to optimize the noise model and the classifier, but with the new version of NLNN they considered the components of the same network. Hence, they optimize the elements simultaneously.

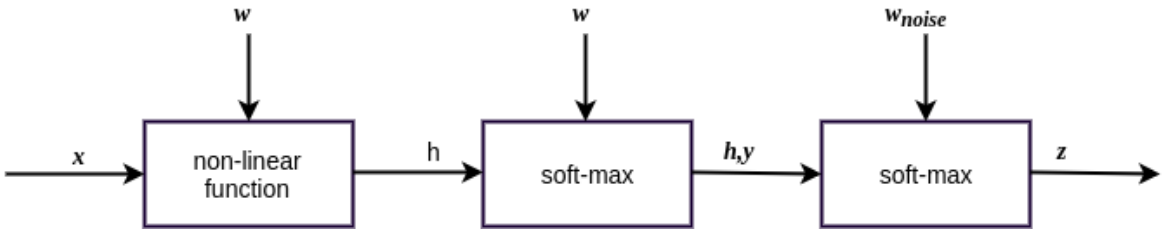

Figure 2.7: Jacob Goldberger noisy label neural network architecture in the training phase [17]

In the testing phase, they removed the extra softmax layer, as they wanted to predict the true labels. In this method, good initialization of parameter of noise adaptation layer is important. They experimented on noisy MNIST dataset and CIFAR-100 for image classification to check the robustness of their method. They outperformed all the alternative noise-robust methods. In our work, we explore the NLNN with EM algorithm for natural language processing task.

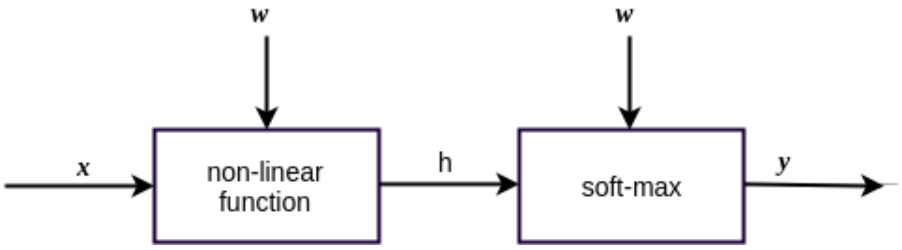

Figure 2.8: Jacob Goldberger noisy label neural network architecture in the test phase [17]

In 2017, [34] Ines Rehbein and Josef Ruppenhofer introduced a method for detection of errors in automatically annotated text. This work reassembles the problem we aim to solve. In this work, they used an unsupervised generative model with human supervision in an active learning framework. There generative model was based on the MACE model proposed by Hovy et al. in [16].

Ines & Josef proposed the following algorithm in [34]. The idea behind using Active Learning is to improve the performance on a new data. They starts with the predictions of a Classifier and then ensemble and learn a Variational inference. They calculate the posterior entropies and perform manual validation with the intervention of human (Active Learning). Finally picks a random classifiers and update the prediction by replacing the classifier's output with the human annotators output. In every iteration they train the variational model. Therefore, they gradually improve the training data.

In this work, they experimented on Natural Language Processing task (POS, NER). They showed that the algorithm can detect errors in articifical annotated text. There are differences between the earlier works and the Noisy Labels Neural Network approach with EM algorithm which has been discussed in the next chapter. One the main significant distention in the previous works is that they have not explicitly modeled noise distribution. In the next section, we will discuss the concept of MTL and related work done in MTL for several Natural language Processing task.

Algorithm 1: AL with variational inference [34]

```
Input: classifier predictions A
for 1 ... n iterations do
  procedure GENERATE(A)
    for i = 1 ... n classifiers do
      T_i ~ Uniform
      for j = 1 ... n instances do
        S_ij ~ Bernoulli(1 - θj)
        if S_ij = θthen
          A_ij = T_i
        else
          A_ij ~ Multinomial(ξj)
        end if
      end for
    end for
    return posterior entropies E
  end procedure
  procedure ACTIVELEARNING(A)
    rank J → max(E)
    for j = 1 ... n instances do
      Oracle → label(j);
      select random classifier i;
      update model prediction for i(j);
    end for
  end procedure
end for
```

2.5 Multi-tasking Learning

Multi-task learning (MTL) is a machine learning technique that aims at improving the generalization performance of a task using other related tasks [23]. Multi-task learning has various applications of machine learning, from natural language processing and speech recognition to computer vision.

2.5.1 Motivation

The motivation of multi-tasking learning comes from various fields. In our life we apply our learning which was acquired from the different homogeneous task. For example, initially a baby only able to recognize faces and then uses the learning technique to recognize other objects [37]. In life we also learn easier problem first which

helps us learning different related complex problem. For example, we first learn addition and subtraction in mathematics then we gradually shift to integration and differentiation.

In machine learning we often see a research problem named transfer learning that concentrate on storing knowledge acquired from one task and apply to another. Similarly, multi-tasking learning can be consider as inductive transfer learning. Inductive transfer introduces an inductive bias, in MTL the auxiliary task introduces that for the main task [37]. This helps to learn more generic representation and hence, more leads to better solutions.

2.5.2 Analyzing MTL

Generally, in the field of Machine Learning we concentrate on optimizing a loss function to achieve an reasonable performance. However, we don't leverage the learning from related task. Therefore, we avoid information that might help us in improving the performance of our desired task. Multi-tasking learning (MTL) thrives on the fact that it can generalize our model more with the help of shared representations between related tasks.

Multi-tasking learning is also known as joint learning, learning to learn , learning with auxiliary tasks etc. Generally, in machine learning problem we optimize a loss function whereas in MTL we optimize two or more loss functions. In [10] Rich Caruana proposed that MTL leverages the domain-specific information provided by the features of the related tasks.

In the next section, We will explain the two different variants of MTL in Deep Learning. We will talk about the auxiliary tasks and the advancement of MTL in the field of Natural Language Processing. In chapter 4, We will discuss more elaborately on the MTL for sequence labelling task.

2.5.3 Hard parameter sharing

Hard parameter sharing is one of the popular approach to multi-task learning in neural networks [37]. The main idea behind hard parameter sharing is sharing the hidden

layers among all tasks, while keeping some task-specific output layers [10]. The following figure [37] portraits the hard parameter sharing.

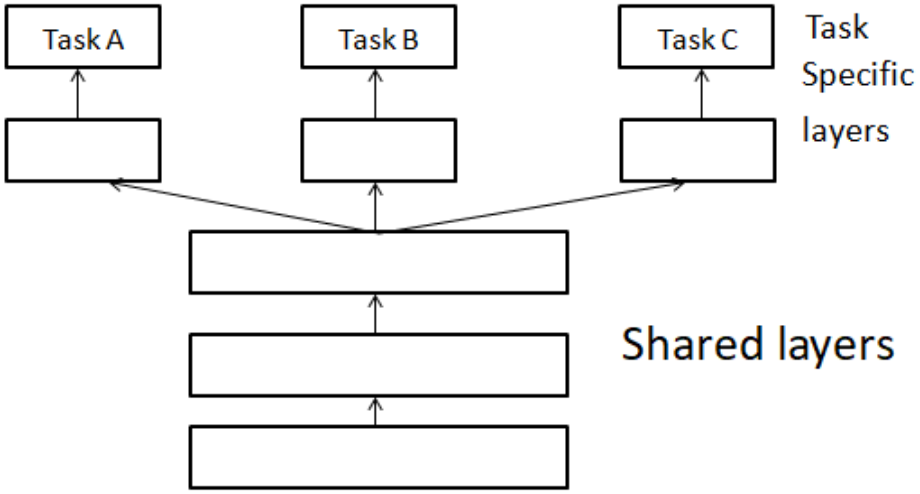

Figure 2.9: Hard parameter sharing for multi-task learning in deep neural networks [37]

The other key aspect of hard parameter sharing is reducing the risk of overfitting. In [3] Jonathan Baxter showed that task-specific parameters are larger prone to overfitting than the shared parameters. Intuitively, more task will help in find a good representation that can capture features for all task. Therefore, it will be more generic. Hence, risk of overfitting will reduce on original task.

2.5.4 Soft parameter sharing

On the other hand, in soft parameter sharing every task has its own model and its own parameters [37]. In order to regularize the distance between the parameters of the model the parameters are kept similar. In [45] used trace norm whereas in [14] used l2 norm for regularization. It might be prone to overfitting. The following figure [37] portraits the soft parameter sharing.

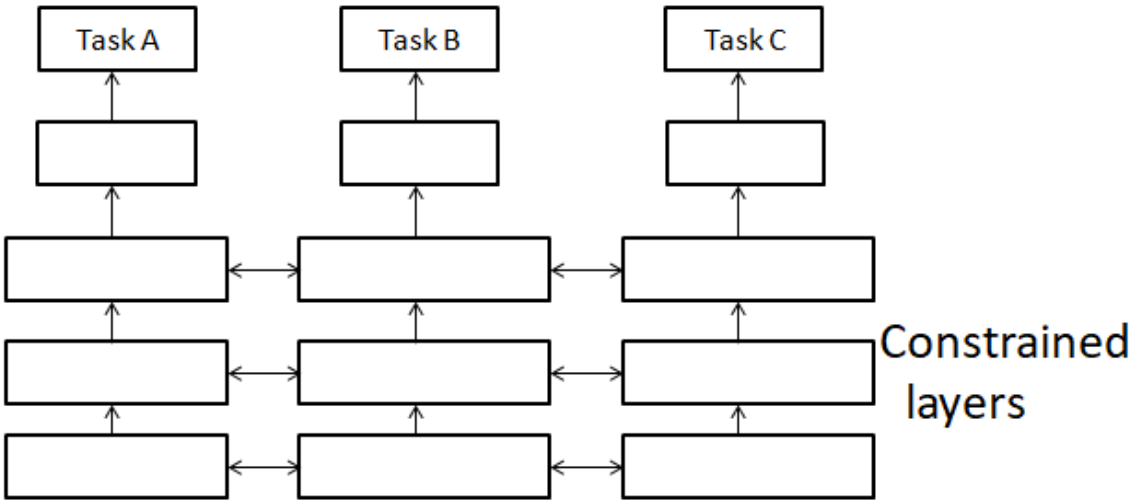

Figure 2.10: Soft parameter sharing for multi-task learning in deep neural networks [37]

2.5.5 Auxiliary tasks

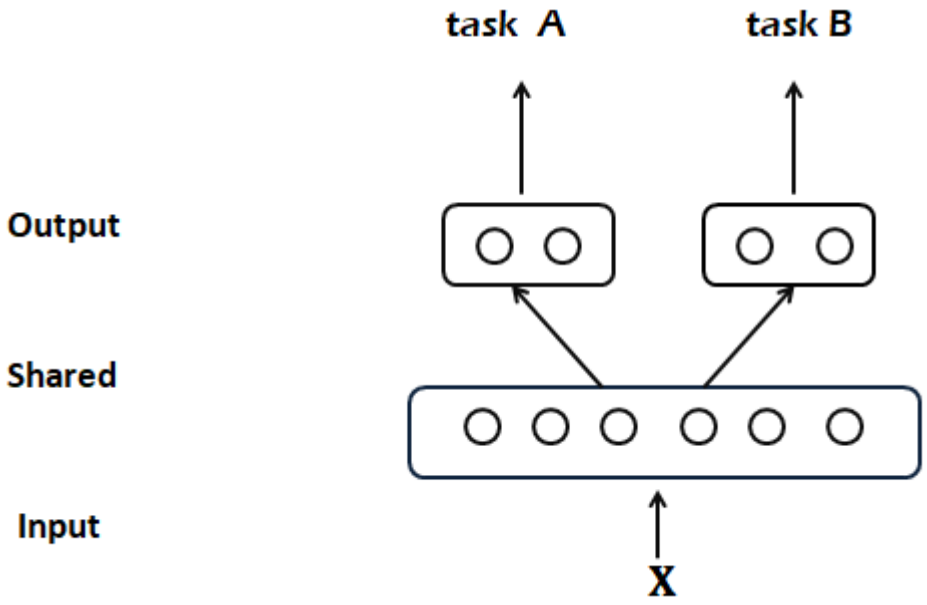

Figure 2.11: Simple Multi-task learning

The whole idea of MTL is based on the fact that how much one task can help another task. Auxiliary task or Related tasks plays a vital role in MTL. Recently,

there has been lot of studies going around related to finding suitable auxiliary tasks for the main task.

MTL suits well in scenarios where multiple tasks are predicted at one time. Therefore, in real life problems (finance, share market forecast, drug discovery) MTL fits well [37]. It has been studied by Ramsundar in [29] stating that increasing the number of tasks performance of MTL increases drastically. Figure 2.11, depicts a simple MTL model where there are shared representation for two different task.

Most commonly we concentrate on a single task, in such scenarios auxiliary tasks can treated as feature engineering. The selection of auxiliary tasks can be quantified by the task similarity, task correlation, task co-variance etc. Recently in [8] Bingel, Joachim, & Anders Søgaard studied on task relationship and how can that help on improving the performance of a MTL model.

2.5.6 Multi-tasking learning in NLP task

In this section, We will discuss some existing work in NLP tasks related to MTL. In the following, We will shed light on some selected NLP task which uses MTL :

Speech recognition In recent times for speech recognition task multi-tasking learning shown some improved results. In [41] stated improve the performance of end-to-end model positioned the intermediate representations as auxiliary at lower levels of a neural network and like-wise phoneme duration and frequency profile can be a auxiliary tasks [4].

Machine translation MTL plays an crucial role in the field of Machine translation. Recently there has been lot of studies related to it such as in [15] and [46] suggested jointly training the decoders and encoders respectively whereas, [18] stated jointly training both encoders and decoders.

Multilingual tasks Similarly in Multilingual tasks, multi-tasking learning improves the performance as jointly training models for differently language. We have seen some improvement in sequence labelling task like NER and POS tagging.

Question answering The main idea is dividing a complex task into simpler task. Then, jointly learning simple task to solve the complex task. Question answering (QA) uses this very idea to learn a end-to-end model. In [11] proposed jointly learning sentence selection and answer generation model, whereas [43] proposed jointly training ranking and reader model.

Information retrieval In information extraction, different relations and roles has shared information. Joint learning different classification and search ranking [22].

Chunking Jointly training chunking as main task and auxiliary task as POS tagging or NER tagging [31]. In chapter 4, We have discussed it in elaborate.

Chapter 3

Noisy Labels Neutral Network

Bekker & Goldberger [7] introduced Noise Labels Neural Network (NLNN) algorithm which addresses the problem of training neural network on noisy labels. They proposed an idea to add an extra layer of noise channel with the neural network architecture. Similar idea was proposed by Sukhbaatar and Fergus's [40] of introducing an extra linear layer after the soft-max layer. They showed the linear layer as the transformation matrix between true labels and noise labels with some strong assumption. However, Bekker & Goldberger [7] modeled the noise generation by a parameter. They [7] assumed that the input features are independent of noise generation in observed labels. During the training phase, NLNN model learns the neural network parameter and the noise distribution of the noise labels using the probabilistic modelling and backpropagation. This chapter illustrates the NLNN model and its training process in details.The variables and equations in this chapter are similar to their original forms in [7].

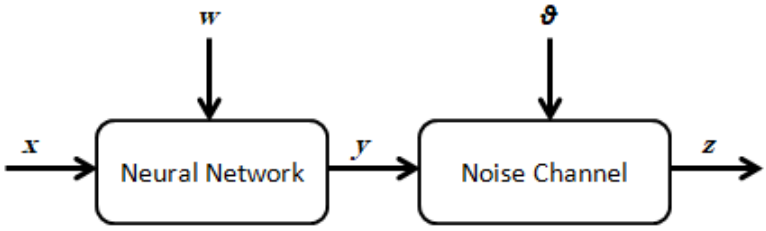

Figure 3.1: Noisy Label Neural Network model

Figure 3.1 describes the NLNN model [7]. The main concept behind the NLNN model is deep neural networks, which is apt to learn from data mixed with inconsistent noise [35]. Hence, it assumes that the output of the neural network will generate hidden true labels given the noisy labels as input.

The output layer of the neural network is a soft-max classifier, which is as follows:

$$p(y = i|x; w) = \frac{exp(u_i^T h + b_i)}{\sum_t exp(u_i^T h + b_i)} \tag{3.1}$$

where u_i is the parameter of the output layer, h is a non-linear function (whose input is x $\therefore$ h=h(x)), b_i is the bias.

In NLNN model, maximum likelihood function is used to find the model parameters. However, maximum likelihood can have infinite numbers of solutions [19]. The NLNN model takes the output of the neural network to estimate the noise distribution. Therefore, the output of the neural network acts as constraints on the model to limit the number of solutions of maximum likelihood. Bekker & Goldberger [7] proposed that Expectation Minimization algorithm is used iteratively to estimate the hidden true labels and the noise distribution in the data.

Figure 3.2: Noisy Channel

Figure 3.2 illustrates a noise channel. The noise channel transforms the hidden true labels into unreliable labels. In this approach the noise generation is modelled as θ which is a confusion matrix between the hidden true labels and the observed noisy labels:

$$\theta(i, j) = p(z = j|y = i) \tag{3.2}$$

It is assumed that we are given n feature vectors $x_1, ..., x_n$ with the corresponding noisy labels $z_1, ..., z_n$ which can be seen as noisy versions of the hidden true labels

$y_1, ..., y_n$ and w is the neural network parameter. The probability of observing noise will be as follows:

$$p(z = j|x; w, \theta) = \sum_{i=1}^{k} p(z = j|y = i; \theta)p(y = i|x; w) \tag{3.3}$$

Therefore, the log-likelihood of the model parameters is:

$$L(w, \theta) = \sum_{t=1}^{n} log(\sum_{i=1}^{k} p(z_t|y_t = i; \theta)p(y_t = i|x_t; w)) \tag{3.4}$$

Our goal is to find the noise distribution θ and the neural network parameter w which maximizes the above likelihood function. In order to learn about the optimal parameters set we use EM algorithm. The EM algorithm is an iterative process for finding the parameters that maximizes the likelihood. The EM algorithm is explained as follows:

In **E-step**:

We estimate $y_1, ...y_n$ true hidden labels from the noisy labels and current parameters (θ,w). The true hidden labels are the output of neural network using the equation below.

$$c_{ti} = p(y_t = i|x_t, z_t; w_0, \theta_0) = \frac{p(z_t|y_t = i; \theta_0)p(y_t = i|x_t; w_0)}{\sum_t p(z_t|y_t = i; \theta_0)p(y_t = i|x_t; w_0)} \tag{3.5}$$

In our problem the noise generated by the artificial tagger, hence the noise was more notorious. We used a smoothing term η to smooth the above equation.

$$c_{ti} = p(y_t = i|x_t, z_t; w_0, \theta_0) = \frac{p(z_t|y_t = i; \theta_0)p(y_t = i|x_t; w_0) + \eta}{\sum_t p(z_t|y_t = i; \theta_0)p(y_t = i|x_t; w_0) + N\eta} \tag{3.6}$$

where, N is number of classes, η is a smoothing parameter $\eta << 1$.

In **M-step**:

We update the confusion matrix θ based on the hidden true labels. It is the confusion matrix between the soft estimate correct labels and the noisy labels.

$$\theta(i, j) = \frac{\sum_t 1_{z_t=j} p(y_t = i|x_t; w)}{\sum_t p(y_t = i|x_t; w)} \tag{3.7}$$

We update the parameters of neural network w using standard back-propagation in order to maximize the log likelihood. Therefore, it estimates the soft true labels.

However, EM algorithm has one drawback that it tends to converge to a local optimum. The above consequence can be overcome with proper initialization. A

potential solution to the challenge of proper initialization can be using well-trained and tuned neural network. The main concept behind using the output of neural network is that a well trained neural network can be more consistent compare to any noisy data set [33]. Based on the output of the neural network we can set the parameters of the EM algorithm.

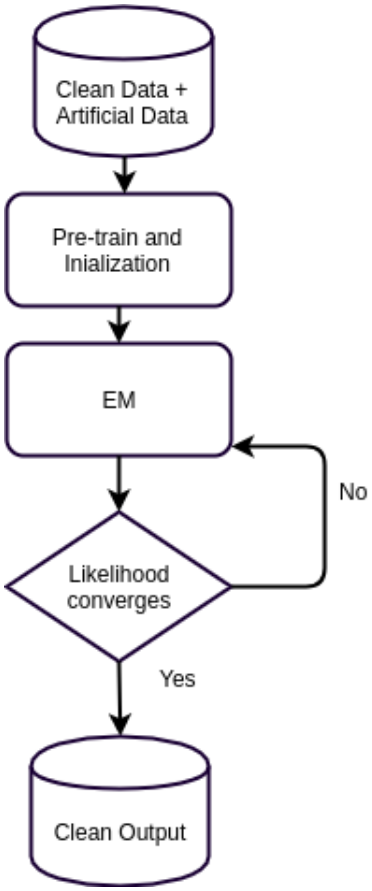

Figure 3.3: Flow Chart of NLNN algorithm

In Algorithm 1, NLNN algorithm is described, where x represents the feature vectors given as input. The weights and biases of the neural network depicted by the parameter w. The output of the neural network is represented by y, that gradually decreases the noise in the data with every iteration. The noise distribution parameter θ gives the likelihood of each true label being given a different label in the given annotations z.

Algorithm 2: Noisy Labels Neutral Network Algorithm

Data: Data points x and corresponding labels z
Result: Predictions y, NN parameter w and noise parameter θ
Initialization: Pre-training NN; Output parameter; initialize θ based on predictions y and original labels z;
while *likelihood converges* **do**
  E-step:
  Estimate true labels c based on parameters w and initialized θ
  M-step:
  Re-compute θ based on c
  Re-train NN to maximize likelihood based on c
end

Bekker & Goldberger in [7] shows the effectiveness of NLNN model. They used uniform noise distribution on MNIST data set (handwritten digits) and the TIMIT data set (continuous speech). They have shown that NLNN model can learn the noise distribution and the neural network parameters w. However, there are many other research questions. In particular, we explored how effective the NLNN model works when we add data with labels produced by the artificial annotator. We also addressed the point that how NLNN model works for the Multi-tasking learning. In the next chapter, we will be discussing about Multi-tasking learning algorithm and modified NLNN with MTL.

Chapter 4

Multi-tasking Learning in Sequence learning

In the field of Machine Learning (ML), we commonly train a single model in order to accomplish a task. In this way, we often attain acceptable results. However, we often ignore the information that can be acquired by learning related tasks [37].

MTL fits naturally to homogeneous tasks. In the field of Natural Language Processing (NLP), there are many heterogeneous tasks, which makes MTL more challenging. However, one task can provide useful features for another task, MTL leverages this idea. Multi-tasking Learning systems are commonly designed to train a single neural network for multiple tasks, using a share representation of features. A recent study, by Héctor Martínez Alonso & Barbara Plank in [2] shows that not all combination of tasks can produce significant results. They proposed that in MTL there are two kinds of tasks one is auxiliary task (POS, Chunk, etc.) and the other is main task (NER, Frames, etc.). This chapter aims to convey an overview of MTL in Deep Learning and the low supervised MTL model. This chapter also demonstrate a possible way to handle noise by setting up a system with the Noisy Labels Neural Network on Multi-tasking Learning. Using the supervised signals to clean labels along with NLNN. The variables and equations in this chapter are similar to their original forms in [1].

4.1 Sequence Labelling Task

In NLP, there are many dependency parsing task, sequence labelling is one of them. Sequence labelling task is a type of pattern recognition problem. In the last decade,

there has been a lot of research about sequence labelling task. Most sequence labelling task were solved using probabilistic methods and statistical model.

Few common methods are Conditional random field, Hidden Markov model etc. Alongside that, there has been a lot of rule based tagger formulation, for example, Brill Tagger for part-of-speech task. In recent times, with the advancement of neural networks, we have seen some improved performance.

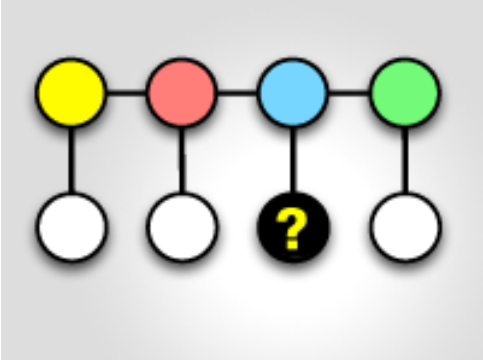

Figure 4.1: Sequence Labelling Task

Figure 4.1, illustrate a sequence labelling task. In this thesis, we have studied how single-task learning and multi-tasking learning work for sequence labelling tasks, such as part-of-speech, chunking etc using Bi-LSTM.

4.2 Supervising different tasks on different layers

The advancement of machine learning algorithm we have seen a rapid assimilation of complex task. Two major factors in MTL are the architecture used for learning and the auxiliary task [37]. In [38] Shen & Sarkar proposed that there are some natural hierarchy among the different tasks, some tasks may get more advantage over other tasks, and not the other way around. Based on this idea Anders & Goldberg [1] proposed multi-tasking learning with low level supervised at lower layers. One way of looking at the concept of supervision in low-layer is pre-processing method. Therefore, the low-level task such as part-of-speech, name entity recognition, etc. are

treated as auxiliary task.

Anders & Goldberg [1] presented an architecture for deep multi-tasking learning. Some important concepts of the architecture are:

Deep recurrent neural network (RNN) It is also famously known as k-layer RNN. In [5], Bengio proposed that deep model can be exponentially more effective than a shallow neural network. Stacking k-RNNs in a way can be more efficient. In Figure 4.2, Anders & Goldberg [1] used 3-layer RNN, where the higher level task is Chunking and the lower-level task is POS tagging.

Describing the MTL model and its architecture.

Figure 4.2: Multi-tasking Learning with low-level supervision

Bidirectional RNN In 1997, Schuster and Paliwal introduced bi-RNN. The main idea was to read the sequence from both forward and backward directions.

$$BIRNN_\theta(x_{1:n}, i) = v_i = h_{F,i} \circ h_{R,i} \tag{4.1}$$

$$h_{F,i} = RNN_F(x_1, x_2, ..., x_i) \tag{4.2}$$

$$h_{R,i} = RNN_R(x_n, x_{n-1}, ..., x_i) \tag{4.3}$$

The current states can not only get the past information but also the future input information, which naturally improves the performance of the neural network. Finally stacking or connecting the hidden layers to provide information to the output layer.

In [1] they leverage the above concepts to build a deep bidirectional RNN. Figure 4.2, describes the architecture proposed by Anders & Goldberg [1]. The intensity of the colour signifying the information passing on to the higher layer. Clearly, in this figure part-of-speech tagging is used as a pre-processing step to chunking.

4.3 MTL meets Noise Labels

As we had discussed earlier, the advancement of MTL in the field of NLP helped to improve and learn several tasks in parallel. For learning new tasks like human being, MTL apply knowledge obtained by related task. Generally, artificial annotators are used to label large data. However, that introduces unreliable labels in the training data. In this section, we will discuss the idea of MTL with Noise labels.

In Single task learning (STL) we train a model on some task A, our goal is to find a good representation for task A. Good representation will produce a model that have a trade-off between bias and variance. However, generally STL can be prone to over-fitting. When there are noise labels in the data, a model that learns to generalizes the data well can perform better. Therefore, MTL can be a handy method as different tasks have different noise patterns [37], it can learn two or more tasks simultaneously. MTL also increases the size of the data as it aggregates all the data of different tasks. Intuitively, MTL can handle noise labels better than STL as it will generalize the representation well.

4.4 NLNN meets MTL

In this section, we will explore the idea of NLNN with MTL. As we had discussed earlier in chapter 3, NLNN can be used to find the noise distributions and also for

estimating the neural network parameters. In MTL, the situation becomes challenging as we have one main task and one or more auxiliary tasks. We intent to find the noise distribution in the training data for all task and clean them simultaneously.

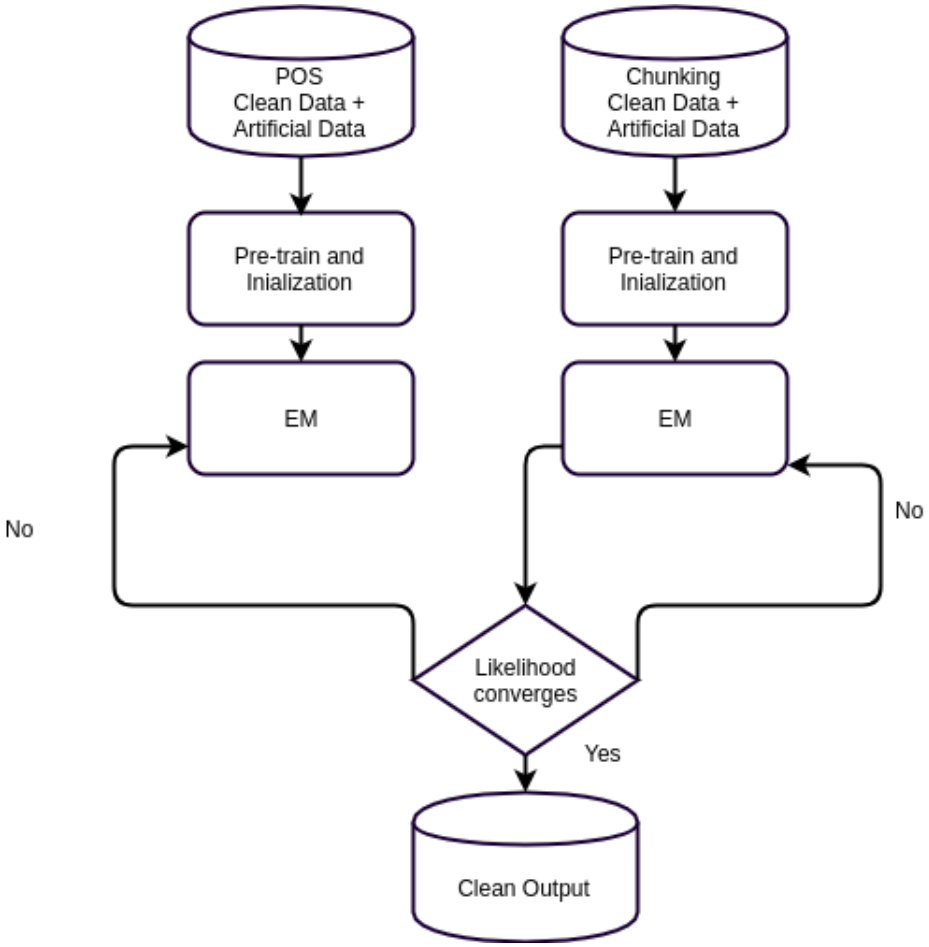

Figure 4.3: Flow Chart of the MTL with NLNN approach

A possible way of solving the problem is to combine MTL with NLNN in such a way that not only can we get a better classifier but also a clean training data. Figure 4.3, demonstrates the NLNN with MTL. Here, the EM algorithm is working separately for two different tasks and finds the noise distribution for both the tasks. In this method, the NLNN will run until the likelihood of the main task convergences. The auxiliary task is providing information (pre-processing step). The core idea behind this setup is to improve the generalization performance. The classifier will learn to

find a good representation of the data. Hence, we improve the overall performance of the classifier for the main task. In the next chapter we will show the experiments we performed and discuss about the performance of NLNN in depth.

Chapter 5

Experiments

In this chapter, we will discuss the different experimental setups and evaluation metrics that has been used to evaluate our work. In the work done by Bekker & Goldberger [7], they performed NLNNs on two tasks: a handwritten recognition task and phoneme classification tasks. They had generated noisy labels from clean labels by injecting noise using permutation and uniform distribution. In our experiments we have used artificial annotator to annotate labels, hereby injecting noise. We will demonstrate how powerful NLNN model can be to counter artificial noise. We will also demonstrate how NLNN model can perform when we vary the size of the clean data.

The goal of the section is to explore the following questions :

- How effective is the NLNN model on POS tagging, when small amount of clean labelled data is added?
- How effective is the NLNN model on POS tagging, when large amount of clean labelled data is added?
- How effective is the NLNN model on Chunking?
- How effective of the NLNN model on Higher level task Chunking with low supervision of low-level task (MTL)?

5.1 Corpus

While experimenting with various Natural Language Processing methods, **Penn Treebank** corpus is used to train and test methods. Penn Treebank corpus contains 9,997 words. It contains human annotated part-of-speech and chunking tags. This corpus is also famously used for Language modelling task. We divided the data set into three

sections 0-20 for training set, 21-22 for development set, and 23-24 for testing set. In our experiments we assumed that Penn Treebank corpus contains noise free labels.

The **North American News Text NANT** corpus comprises of news text, formatted using TIPSTER-style SGML markup. It is a collection of text in English from news-wire and newspaper sources in America. It contains approximately 1 billion words or more. We were required to do pre-processing in order to transform the NANT into CoNLL data format. The pre-processing steps involved forming sentences using Stanford parser[1], removing short sentences and then, transforming them into CoNLL format. NANT contains news corpus which are L.A.Times/Wash.Post, Reuters General News, Reuters Financial News, Wall Street Journal, New York Times Synd. In this work, we used L.A.Times/Wash.Post News Corpus to make task more challenging.

5.2 Evaluation metrics

We have used F_1 score to evaluate performance. F_1 score can be interpreted as a weighted average (harmonic mean) of the precision [2] and recall [3]. The upper bound of an F_1 score is 1 and lower bound is 0.Precision and recall has equal contribution to the F_1 score. The formula for the F_1 score is:

$$F_1 = \frac{2 * (precision * recall)}{(precision + recall)} \tag{5.1}$$

We have used 'macro' F_1-score which calculates metrics for each label, and find their un-weighted mean. This method does not take label imbalance into consideration. We used scikit-learn [4] to calculate the macro F_1score.

5.3 Artificial Annotator

5.3.1 Brill Tagger

In 1993, Eric Brill had introduced a ruled based part-of-speech tagger. The tagger induced grammar directly from the training corpus [24]. He introduced a system that

[1] `https://nlp.stanford.edu/software/lex-parser.shtml`

[2] the number of true positives T_p over the number of true positives and the number of false positives F_p

[3] the number of true positives T_p over the number of true positives and the number of false negatives F_n

[4] `http://scikit-learn.org/stable/`

can learn lexical/morphological and contextual information from the training corpus and can also deduce the best possible POS tag for a word [24]. It can learn without any intervention of human or expert knowledge. It is an "error-driven transformation-based tagger" [9]. It is a supervised learning method which tries to minimize the error and it is also a transformation-based process, in the sense that a tag is assigned to each word and changed using a set of predefined rules. After the training, the tagger can be used to annotate with part-of-speech tags to any corpora.

5.3.2 Senna

It is a software tool which performs various Natural Language Processing predictions such as name entity recognition (NER), part-of-speech (POS) and chunking [30]. SENNA has simple architecture and offers state-of-the art performance. Hence, it is fast and accurate. It is written in ANSI C and requires about 200MB of RAM [31].

5.4 Experiment 1

In the field of computational linguistics and natural language processing, part-of-speech tagging (POS) is one of the fundamental problems. The part-of-speech tagging is important as it conveys a lot of information about a word and its neighbouring words. POS tagging is useful for several other computational linguistic problems such as name-entity tagging, chunking, etc.

Part-of-speech tagging is a challenging task as some words can have multiple tags depending on the context. There has been a lot of research dedicated to solving it, mostly using probabilistic methods, statistical model and rule-based methods. In the last decade, we have seen some improved performance, mostly neural network based POS tagger with word embedding and character embedding [28].

5.4.1 Methods

The training data is a mixture of clean labelled data [5] (PTB) and the data produced by the artificial annotators (Brill Tagger). The Noisy Labels Neural Network was trained on the above data. Initially, we used Bidirectional Long Short-Term Memory with word embedding to train on the mixed noise data for 10 epochs. The output of the neural network is used to initiate the EM algorithm and run until it converges.

[5]Labelled under the supervision of human or expert knowledge

The EM algorithm took on average 4 iterations to converge to a local optimum.
The goal of this thesis is to clean training data, hence produce a better classifier. In each iteration of NLNN, the standard backpropagation produces a model that tries to generalize the data. Intuitively, in each iteration the model is also learning from the noise in the data. Hence, it might produce a training output which might corrupt the clean labelled data shown in the figure 5.1.

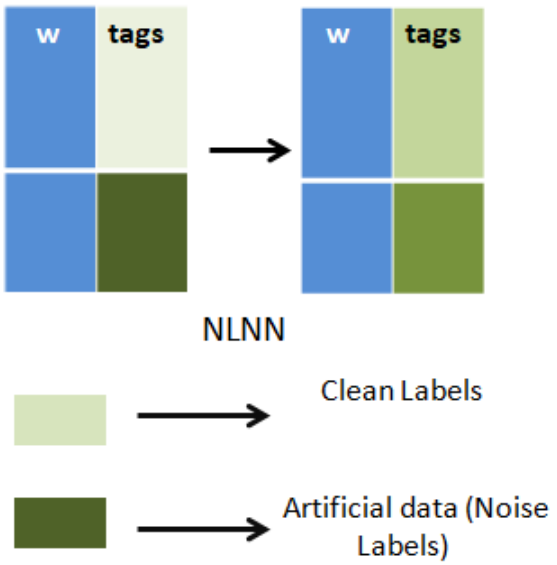

Figure 5.1: NLNN after one step, noise amount is shown by using different shade of colour green where dark shade implies highly noise labels

In order to avoid such scenarios we came up with a modified version of NLNN, where we kept replacing the clean data part with the original clean data in every iteration. In figure 5.2, shows the updated flow of NLNN in after every iteration. The color intensity suggests the noise percentage in the data. Dark green shade suggests artificial data (mixed with noise labels) and light green shade describes the clean data. The part-of-speech tagging is a complex task and it has 53 different tags which are as follows: AUX, AUXG, CC, CD, DT, EX, FW, IN, JJ, $JJ|DT$, JJR, JJS, LS, MD, NN, NP, PP, NNS, NNP, NNPS, PDT, POS, PP\$, PRP, PRP\$, RB, RBR, RBS, RP, SYM, TO, UH, VB, VBD, VBG, $VBG|NN$, VBN, VBP, VBZ, WDT, WP, WP\$, WRB, $PRP|MD$, :,), (, ., ,, ”, “, \$, # .

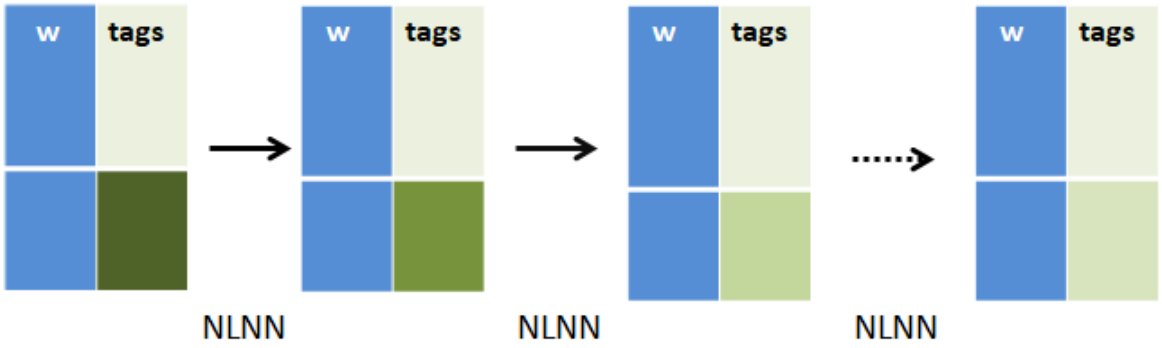

Figure 5.2: Flow of the NLNN, iteration wise

5.4.2 Results

In recent times, massive growth in the amount of data is exciting. However, the size of the clean labelled data may vary. There are two ways of looking at the problem which are : 1) How much artificial data we can add so that the performance of NLNN stays better than the clean human annotated data? (Here clean data is fixed) 2) How much clean data we can replace with artificial data so that the performance of NLNN stays better than the BiLSTM ?

The idea of In order to check the efficiency of the modified NLNN, we have run the following experiments with varied sizes of the clean data:

Replacing clean data with artificial labelled data

In this experiment, we wanted to explore the fact how efficiently NLNN can handle noise when we keep replacing clean data with artificial labelled data and also keeping the training data size fixed. We compare the performance of NLNN with the base Bi-LSTM model. We have used Brill tagger to tag few sections of Penn Treebank dataset to generate artificial labelled data. Table 5.1, shows that modified NLNN improves the performance over NN. The performance of NLNN shows

Table 5.1: F_1-score of NLNN and NN on POS tagging

F_1 score		
Artificial data	NN	NLNN
10%	92.89	93.26
20%	92.07	92.93
30%	90.85	92.60
40%	88.46	91.86
60%	87.37	88.19

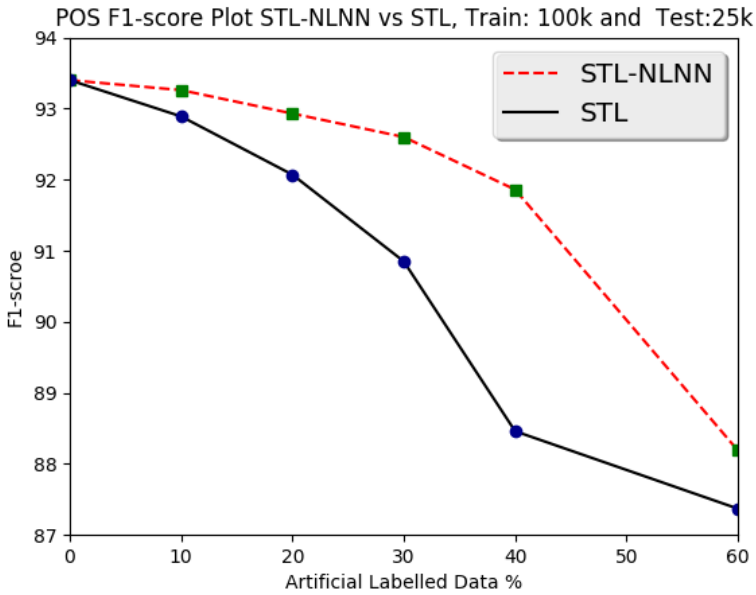

Figure 5.3: Performance of STL-NLNN and STL-NN for POS tagging

This is one of the initial experiments which motivated us to check the robustness of NLNN, when we add artificial labelled data to our clean labelled data. In the following experiments we shift towards adding artificial labelled data and performed NLNN on those data.

Fixed Small clean labelled data and additional artificial data

In this experiment, in order to train my model we have used 100k words with clean labels from Penn Treebank and have added the artificial data accordingly. we have manually mixed the data-set. Table 5.1, shows that modified NLNN improves the performance. However, as the proposition of the artificial data increases, we can see a clear dip in the performance.

Table 5.2: F_1-score of NLNN and NN on POS tagging with small clean data

F_1 score		
Artificial data	NN	NLNN
10%	93.57	93.59
20%	93.33	93.64
30%	93.48	93.65
50%	92.74	93.47
66%	92.85	93.33
75%	92.60	93.00
83%	91.91	92.33

In figure 5.3, we can see that NLNN approach improves the performance as we add artificial data till 30%. The EM algorithm is able to recover until the noise percentage raises due to the increase of artificial data. We can clearly see that NLNN is more stable than NN.

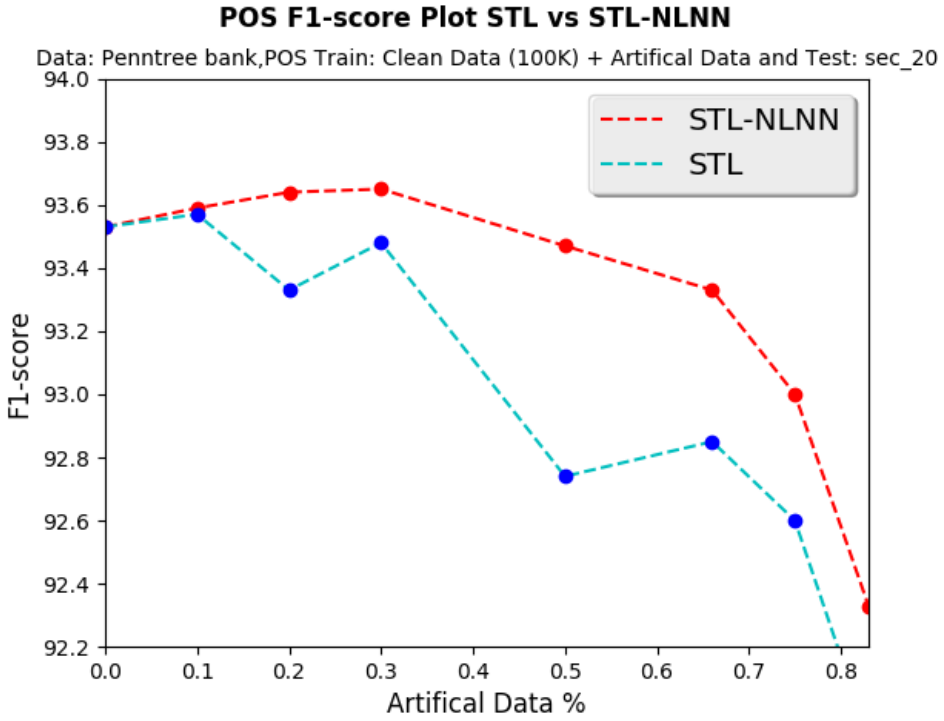

Figure 5.4: Performance of NLNN and NN for POS tagging with small clean data

Fixed Large clean data and additional artificial data

In this experiment, to train our Bi-LSTM model we have used 1 million words with clean labels from Penn Treebank and have added the artificial labelled data (NANT) accordingly. We have manually mixed the dataset. The idea behind increasing the size of the data is to explore the robustness of NLNN to eminent amount of noise. Table 5.2, shows that modified NLNN improves the performance. However, as the artificial data proposition increases, we can see a clear dip in the performance.

Table 5.3: F_1-score of NLNN and NN for POS tagging on large clean data

F_1 score		
Artificial data	NN	NLNN
10%	95.85	95.94
20%	95.74	96.01
30%	95.69	95.90
50%	95.14	95.20

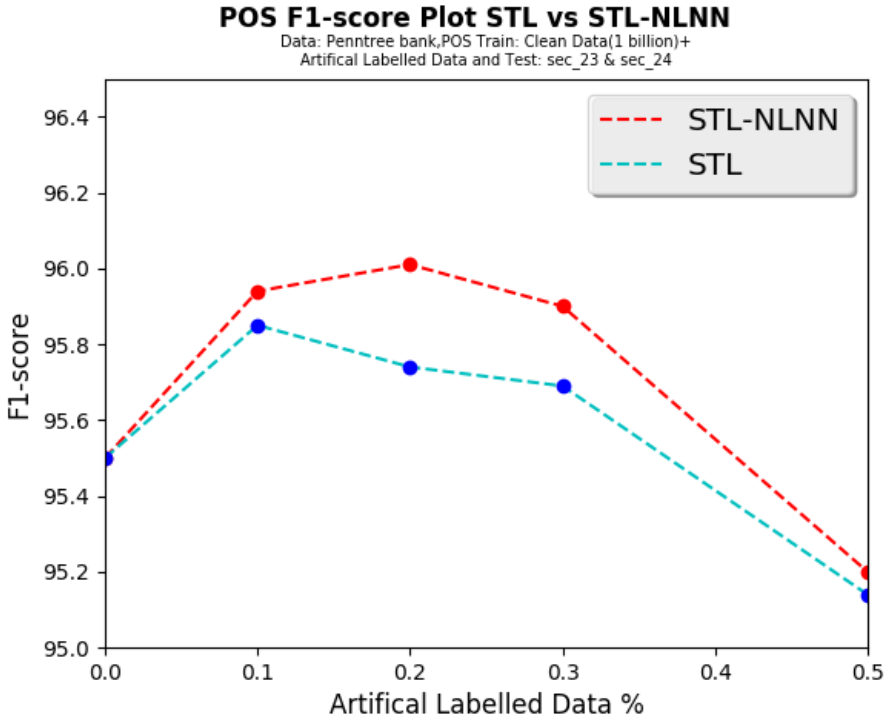

Figure 5.5: Performance of NLNN and NN for POS tagging with large clean data

5.4.3 Analysis and Examining

In both the above experiments, we have seen that NLNN have outperformed Bi-LSTM neural network. However, if we examine the result in depth we can clearly see that NLNN fails to recover the noise after adding 30% of artificial data.

The probable reason behind the failing of NLNN can be as follows:

- After adding 30% of artificial data, the noise labels increases so much that NLNN fails to detect the noise labels. In NLNN, in each iteration, we have used BiLSTMs NN to find the hidden true labels. The Bi-LSTM tries to generalize the data and find a good representation. Artificial noise labels are harder to detect in compared to uniform or random noise. Due to the increase in number of artificial noise labels NN rather than replacing the noise labels with correct labels, finds a good representation. Hence, NLNN fails to recover noise.

- Part-of-speech tagging is a challenging problem. The classification task has 53 numbers of class. Each word can have more than one tag. Hence, NLNN for POS tag after adding 30% of artificial data fails to recover.

In the next section, to examine our second point we have experimented with a more simpler task **chunking**. The motive behind the next experiment is to check whether NLNN can handle noise for more simpler task.

5.5 Experiment 2

Chunking is also known as shallow parsing. It is used for entity detection. Chunking is vastly used in natural language processing. It is a prepossessing step for other tasks like NLU (natural language understanding), speech generation etc.

Chunking is a higher level task in compared to part-of-speech tagging. Chunking is basically grouping part-of-speech tags in a sentence. It is useful as it describes the structures of a sentence. Part-of-speech (POS) is a word-level task as it deals with words, conversely, chunking is a phrase level task.

The chunking tags are as follows: 'b-np', 'b-pp', 'i-np', 'b-vp', 'i-vp', 'b-sbar', 'o', 'b-adjp', 'b-advp', 'i-advp', 'i-adjp', 'i-sbar', 'i-pp', 'b-prt', 'b-lst', 'b-intj', 'i-intj', 'b-conjp', 'i-conjp', 'i-prt', 'b-ucp', 'i-ucp', 'i-lst'. Figure 5.5, portraits an example of chunking [39] , where the each of the large boxes illustrates a chunk .

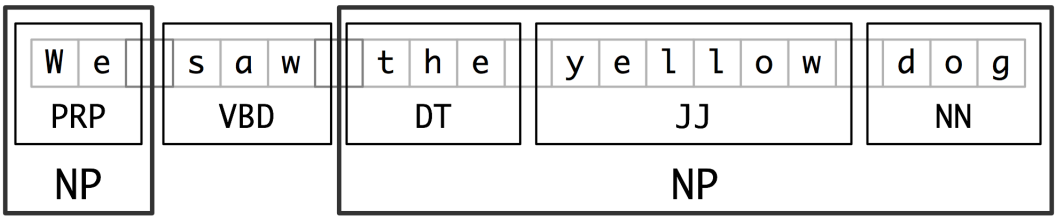

Figure 5.6: Chunking-segmentation

5.5.1 Methods

The training data is a mixture of clean data (Penn Treebank section 15-18) and the data produced by the artificial annotators (Senna). The Noisy Labels Neutral Network was trained on the above data for the chunking task. Initially, we had used Bidirectional Long Short-Term Memory with word embedding to train on the mixed noise data for 10 epochs. The EM algorithm had used the output of the neural network to estimate the noise distribution and true hidden labels. The NLNN runs until the EM algorithm converges to a local optimum. The EM algorithm took an average of 4 iterations to converge for various proportion of artificial data in the training corpus.

5.5.2 Results and Analysis

In table 5.4, shows that modified NLNN improves the performance. We can clearly see the NN is more unstable in the presence of artificial data. NLNN still recover from noise in presence of 50% artificial data. Therefore, for simpler task NLNN works well.

Table 5.4: F_1-score of NLNN and NN on Chunking

F_1 score		
Artificial data	NN	NLNN
10%	95.50	95.54
20%	95.57	95.63
30%	95.46	95.67
40%	95.50	95.67
50%	95.56	95.68

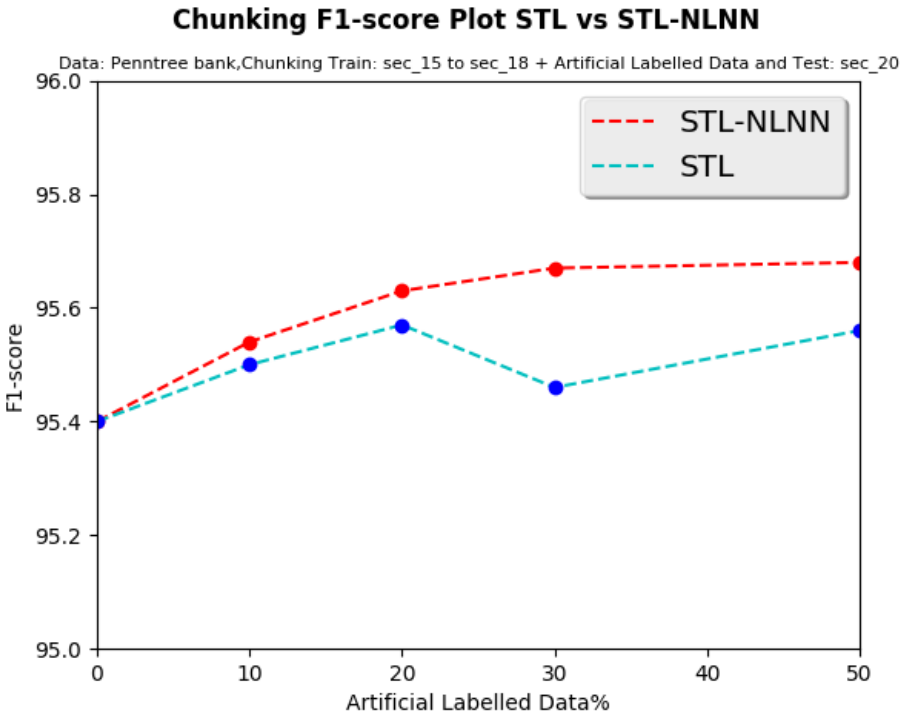

Figure 5.7: Performance of NLNN and NN for Chunking

The positive aspect of this is that NLNN can recover from noise. However, as the artificial data increases we see NLNN getting constant. The consistence might be

because by adding more data the Bi-LSTM itself cannot learn much. Hence, NLNN is getting saturated and can't clean more.

To proceed further with the idea of cleaning noisy labelled data. We can take the help of the information provided by the related task. This idea then takes us to multi-tasking learning. However, there are two aspects of using multi-tasking which are as follows:

- We can use MTL just for providing more information from an auxiliary tasks for a main task,
- We can clean two or more task simultaneously using NLNN and MTL.

In the next experiment we are going to explore how we can handle unreliable labels for two or more task simultaneously.

5.6 Experiment 3

In reality getting clean human annotated big data is nearly impossible. Therefore, cleaning two or more task simultaneously will be a more useful task. Most of the natural language processing task are heterogeneous in nature. Multitasking learning leverages the fact that one task can provide information for other. While training a model using MTL we can clean both auxiliary task and main task simultaneously using NLNN. In MTL, the output of POS tagging is from 1^{st} layer and for chunking from 3^{rd} layer.

5.6.1 Method

In this experiment, we have two data sets, one for part-of-speech tagging and another for chunking. The training data for POS is a mixture of Penn Treebank and the data with labels produced by Brill Tagger. Similarly for chunking the training data is a mixture of Penn Treebank and the data with labels produced by Senna.

The Noisy Labels Neutral Network was trained on the above data. Initially, we have used deep MTL with low-level tasks supervised model [1] with word embedding to train on the mixed noise data for 10 epochs. We did MTL training for POS+Chunking, with chunking being the main task (higher level) and POS as the auxiliary task (lower level). The output of the neural network is used to initiate the EM algorithm and run until it converges. The EM algorithm takes around 4 iterations

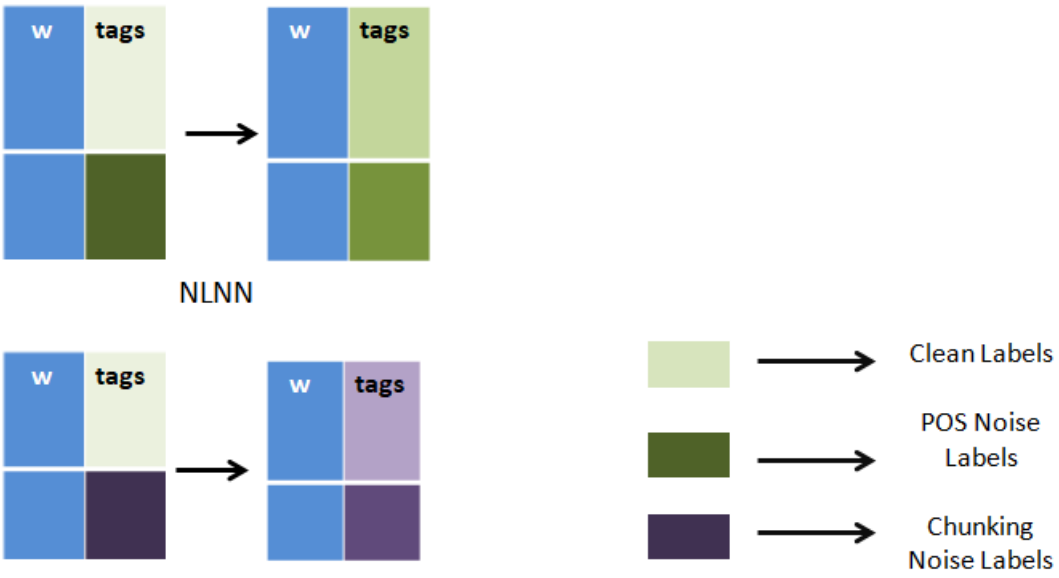

Figure 5.8: Flow of the NLNN, after an iteration, noise amount is shown by using different shade of colour green and violet where dark shade implies highly noise labels

on average to converge to a local optimum. While implementing the MTL + NLNN, we used the modified version of NLNN. Figure 5.7, portraits the problem that neural network tends to learn from the noise. Hence, we kept replacing the clean data part with the original clean data in every iteration. Figure 5.7, shows the whole flow of NLNN with MTL just to eliminate noise labelled data from the clean part.

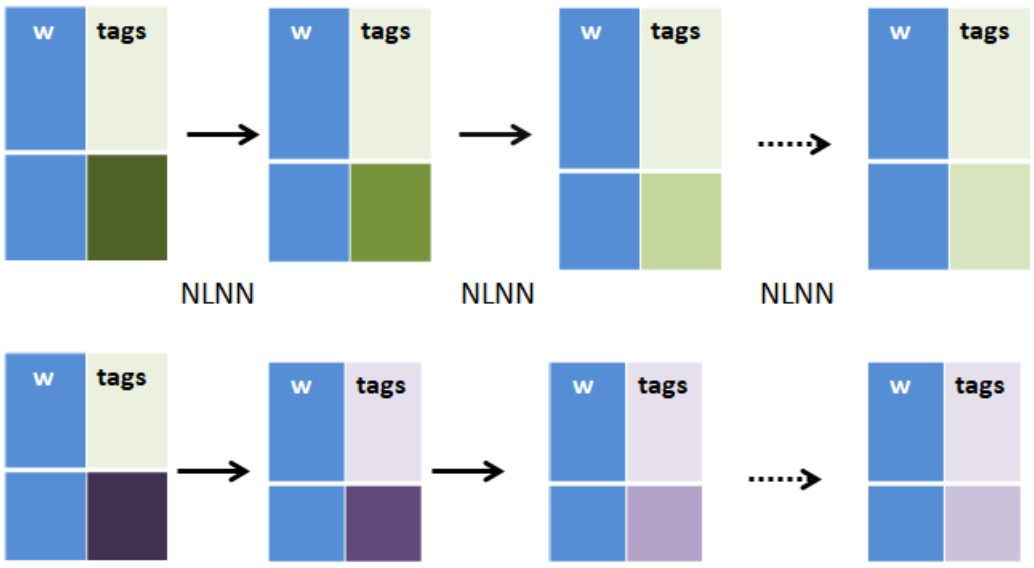

Figure 5.9: Flow of the NLNN with MTL, iteration wise

5.6.2 Results and Analysis

Table 5.4, shows that modified MTL-NLNN outperformed both MTL-NN and STL-NLNN. We can clearly see that MTL-NN is more unstable in the presence of an artificial data in compared to both other results in figure 5.9.

Table 5.5: F_1-score of MTL-NLNN ,MTL-NN and STL-NLNN on Chunking

F_1 score			
Artificial data	STL-NLNN	MTL-NN	MTL-NLNN
10%	95.56	95.48	**95.77**
20%	95.61	95.76	**95.88**
30%	95.65	95.74	**95.84**
50%	95.66	95.72	**95.74**

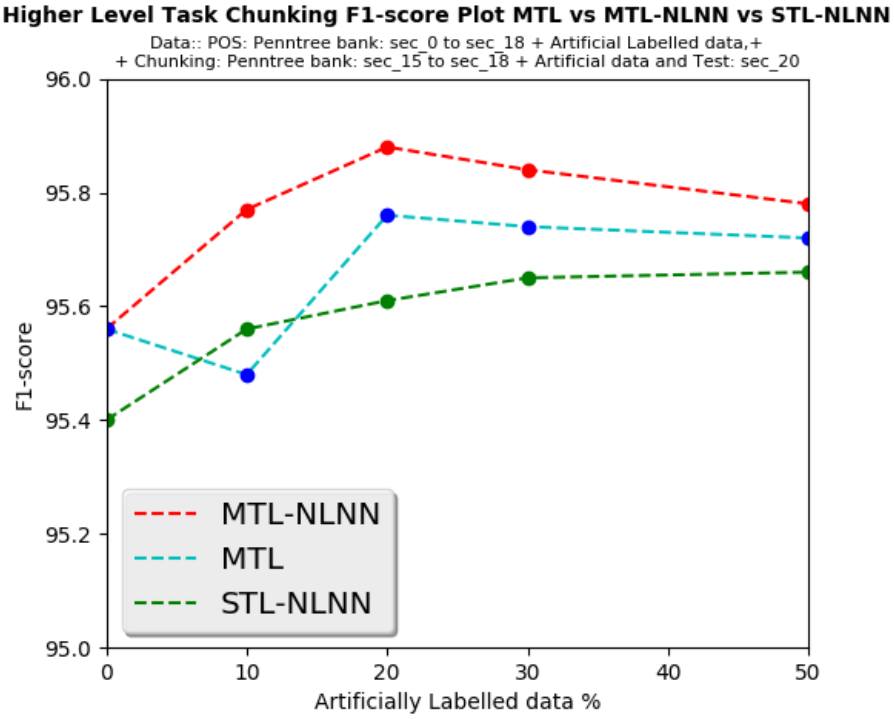

Figure 5.10: F_1-score of MTL-NLNN, MTL-NN and STL-NLNN for higher-level task chunking

The MTL-NLNN for chunking gives the state-of-art performance. However, as we study the figure 5.9, closely we see a dip in the performance of MTL-NLNN as we keep adding artificial data over 30%.

If we compare the results of STL-NLNN and MTL-NLNN, in the STL-NLNN after adding 30% of artificial noise, an increase in F_1-score can be observed, whereas for MTL-NLNN there is a dip in F_1-score. The reason could be that the noise in the auxiliary task is not appropriately recovered by NLNN. In order to study that we look at the performance of MTL-NLNN on POS tagging. Table 5.5, contains the performance of MTL-NN and MTL for inner layer task POS tagging.

Table 5.6: F_1-score of MTL-NLNN and MTL-NN for lower-level task POS Tagging

F_1 score		
Artificial data	NN	NLNN
10%	95.50	95.54
20%	95.57	95.63
30%	95.46	95.67
40%	95.50	95.67
50%	95.56	95.68

After adding 30% of artificial data, the MTL-NLNN for POS tagging fails to recover the noise. This might be a probable reason why the performance of MTL-NLNN drops after adding 30% of artificial data. In MTL, the goal is to find a good representation of the data. The noise for POS tagging effects the performance of NLNN for chunking.

It is interesting to observe which factors and words are responsible for such behaviour. We inspect the human tagged POS tagging (Penn Treebank) and also the final output of the NLNN to get an insight. In [12], Dickinson, Markus, and W. Detmar Meurers did a study on the Penn Treebank annotation. The explicitly mentioned that 50 words are incorrectly tagged as DT (Determiners). For example, tagging "half" as DT and conversely, the word "the" is tagged as NNP.

Word	**True**	**Predicted**
the	nnp	dt
half	dt	nn

They also found 2466 cases of hyphenated words tagged as nouns preceding nouns. The above table gives an overview of that we have discussed above. This might be a valid reason behind NLNN failing to detect noise as there are human annotation errors.

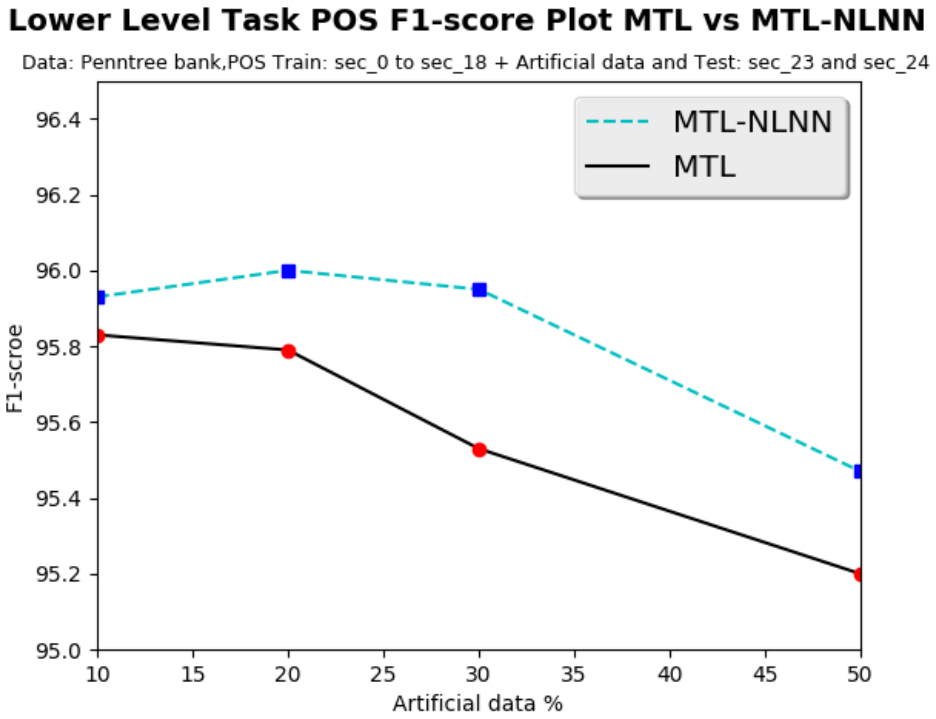

Figure 5.11: F_1-score of MTL-NLNN and MTL-NN for lower-level task POS Tagging

5.6.3 Examining the strength of NLNN

In this section, we are going to measure the strength of NLNN by comparing with the ground truth. The key factor which are responsible for the performance of NLNN are as follows:

- Percentage of artificial data,
- Noise distribution in the artificial data.

In order to examine the strength of NLNN we did the following experiment. In this experiment, we made three separate data set which are as follows:

1. Clean 100k + additional clean labelled data (Human annotated Penn Treebank data)
2. Clean 100k + additional artificially labelled data (Brill Tagger on Penn Treebank data)
3. Clean 100k + additional artificially labelled data (Brill Tagger on North American data)

In Machine learning methods, it is a prevalent thing to say "gold standard labels" when a training data is labelled under human expert supervision. Our first data-set is therefore a "gold standard" labelled data, as it is fully annotated under the supervision of human experts. Bi-LSTM with word embedding was trained on the data to set a bench mark for performance. We trained NLNN on the other two data sets as they contain the artificially labelled data. The motive behind training NLNN on two distinct data set is to make compare the performance between a clean text and TIPSTER-style SGML markup text, which is quite similar to real life unstructured text.

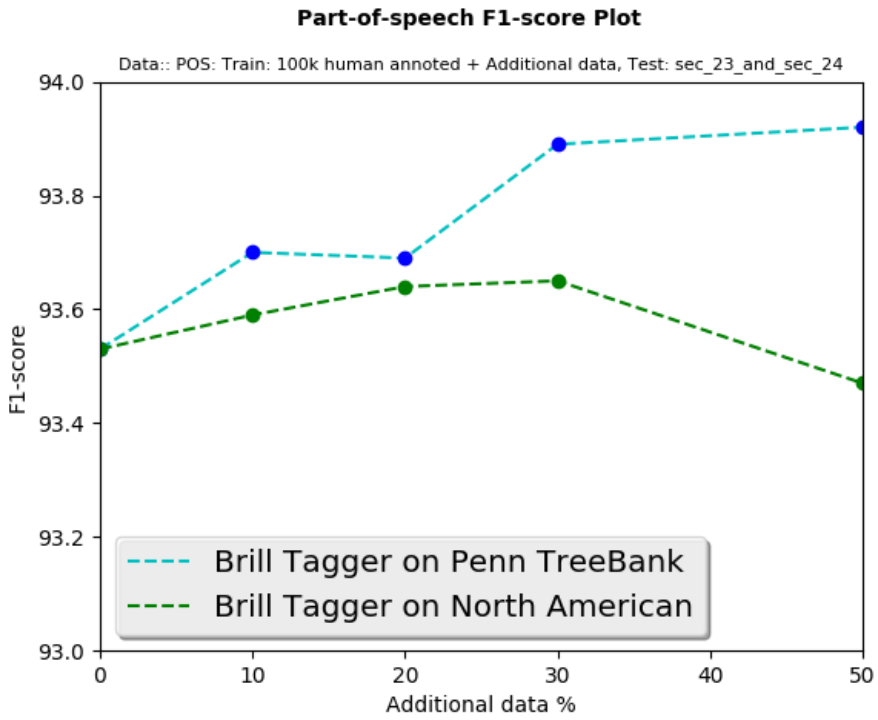

Figure 5.12: Performance of NLNN on artificially tagged for POS tagging

There is a difference in performance when the additional data changes from one corpus to another. We used brill tagger to label the clean text of Penn Treebank whereas for North American Text Corpus first we had to clean the text, use Stanford parser as sentence separator and finally used brill tagger to label the words.
In figure 5.11, we have evaluated the robustness of NLNN to the noisy training data. The NLNN on additional clean text with that artificial labelled outperformed NLNN

on additional raw text with artificial labelled data. The reason behind the performance might be the text Penn Treebank (PTB) is more clean, which means that there is no noisy feature. Another reason might be that the test corpus is annotated under human supervision (Penn Treebank) and while training on similar corpus might have helped the result (context-sensitive).

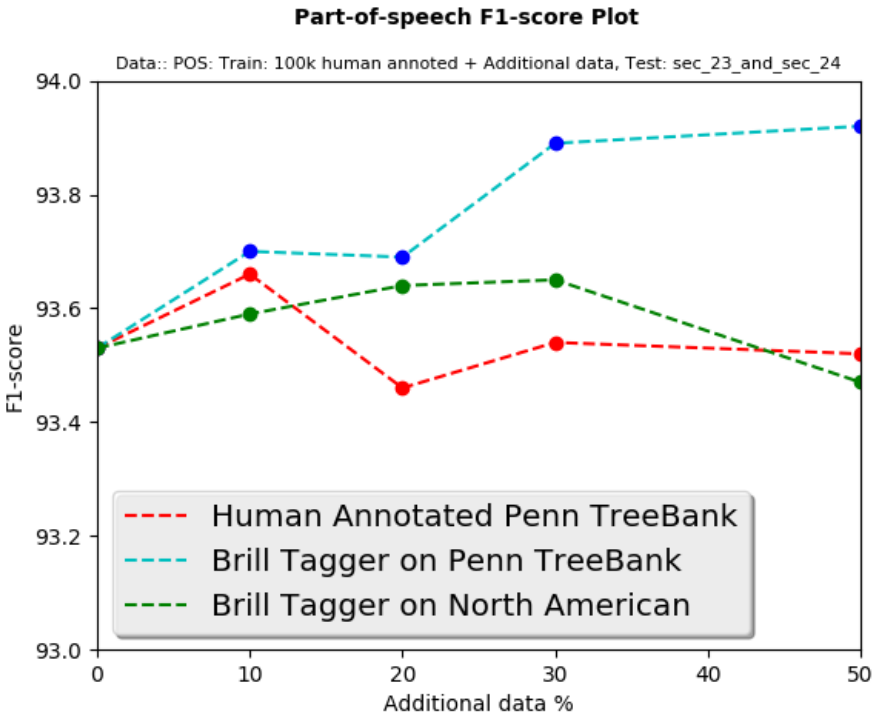

Figure 5.13: Performance of NLNN and NN on different sets of additional data for POS tagging

Table 5.6, shows the performance of NN on fully human annotated labelled data and compared with the above experiment. It is interesting to study that the NLNN on additional clean text with artificial labelled outperformed both the other methods. The reason why NLNN is better than NN is because NLNN can also clean the human annotation error.

Table 5.7: F_1-score of NLNN and NN on different sets of additional data for POS tagging

F_1 score			
Additional data	Gold Standard	Artificially-Labelled PTB	Artificial-Labelled NANT
10%	93.66	93.59	**93.70**
20%	93.46	93.64	**93.69**
30%	93.54	93.65	**93.89**
50%	93.52	93.47	**93.92**

Analyzing the figure 5.12, NLNN can clean the noisy labels present in the training data if the training data is clean text. There is a drop in performance when the artificially labelled has noisy features along with noisy labels. To extend the above experiment we performed another investigation for MTL on the Clean 100k + additional artificially labelled data (Brill Tagger on Penn Treebank data) for POS task and Chunking on a mixture of Penn Treebank and the data with labels produced by Senna.

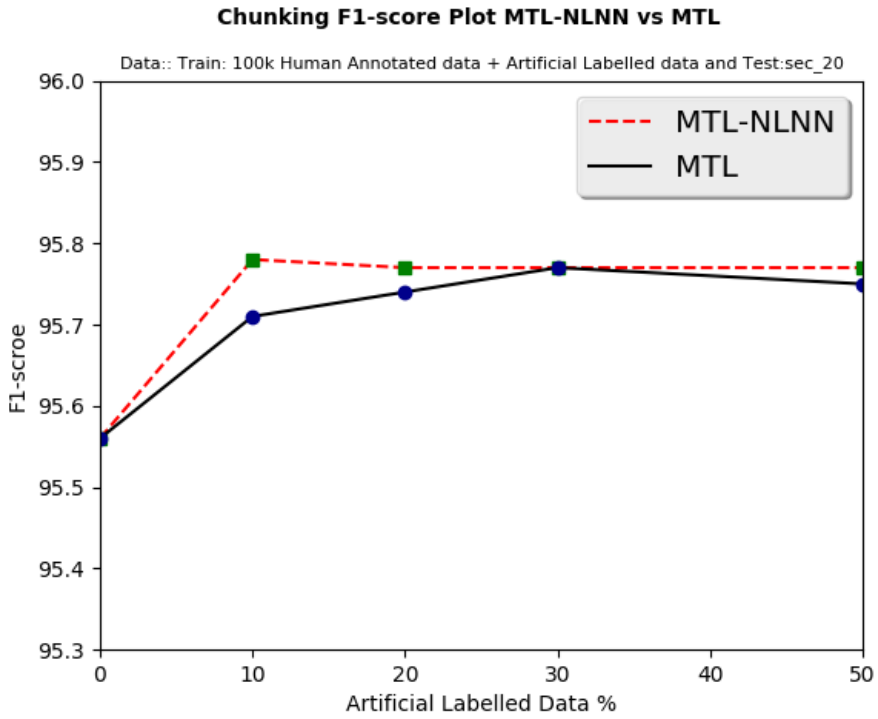

Figure 5.14: Performance of MTL-NLNN and NN on Higher Level task Chunking

We can clearly see almost the same performance of NLNN with a small size of training data for POS tagging for Chunking. In this experiment, the performance of

NLNN and NN are same. So, either the task is saturated as we also have to consider the fact that the NLNN has outperformed the performance of the alternative state-of-art methods for Chunking.

5.7 Discussion

In the above experiments our goal was to handle noise incorporated by automatic artificial labelling. NLNN can handle noise with unknown distribution.

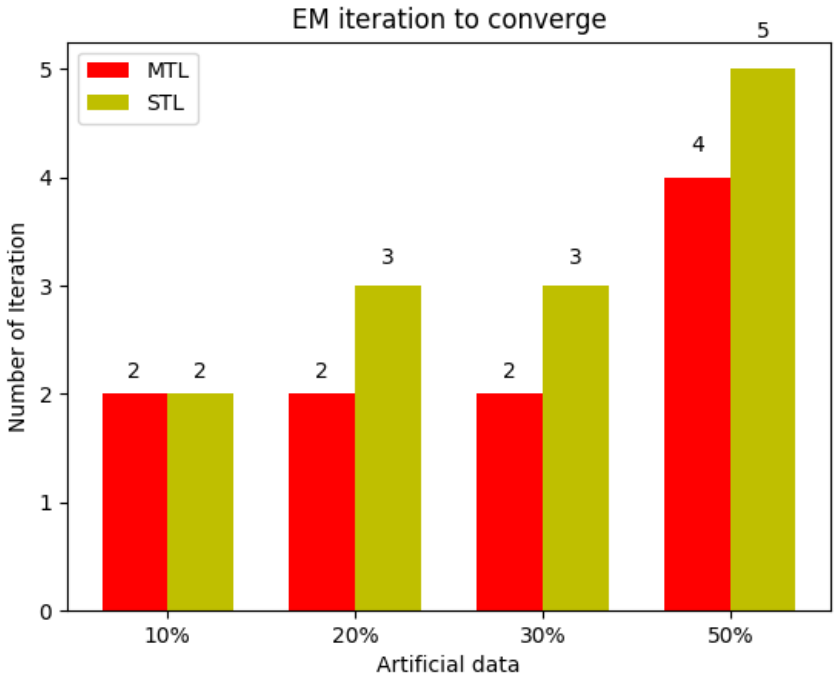

Figure 5.15: EM algorithm convergence rate

There are few important factors involved in the performance of NLNN, which are as follows:

- **Expectation and Minimizationn:** As discussed earlier, we pre-train our neural network model on partial noisy training data to initialize the parameters of EM. EM algorithm is a greedy procedure [7] and an iterative method.

  It is sensitive to the initial point. It can behave notoriously and mischievously for poor initialization. The convergence of EM algorithm depends on the size of the dataset, as we increase the data size it will take more numbers of iteration.

Apparently, for NLNN-MTL, the EM algorithm converges quickly. It might be because the sharing of intermediate representations allows EM to converge faster.

- **Sensitivity to noise features:**

  NLNN tends to work well in handling noise labels. In contrast, it is sensitive to noise labels. The NLNN is more effective on clean text than on raw text. The Bi-LSTM is also context-sensitive. Hence, cleaning the training data depends on various other factors such as handling OOV, unstructured sentences.

The F_1 score for POS tagging reduces as we increase the size of artificially labelled raw text. In contrast, for Chunking the NLNN is not that sensitive and performs consistently. The reason might be that granularity of POS tagging is more than that of Chunking task. Perhaps it is also possible that better tuning of Bi-LSTM neural network could improve the performance on POS tagging. Noise in labels due to artificially annotated is different than artificially adding random noise. The animosity increases as there is no such patterns of noise in artificially labelled data. In our experiments, we used a subsection of NANT which is L.A.Times/Wash.Post News Corpus, however, intuitively, using Wall Street Journal (WSJ) data would have improved the performance, as Penn Treebank is a subsection of WSJ.

An appealing idea is to use MTL with modified NLNN pushed the scope to clean multiple training data with different tasks simultaneously. It helped to improve the performance of Chunking. However, dis-balance in the granularity of different tasks may effect the performance.

In search of an answer to why the NLNN reduces its robustness to noise for POS tagging, we performed error analysis and we found that there are some errors in training and test datasets made by human experts. It might be one of the reasons why modified NLNN failed to improve after inclusion of certain percentage of artificial data.

To conclude, it can be rightly said that NLNN with a better neural network and clean input text can be used to clean labels and can help to achieve a better classifier.

Chapter 6

Conclusion & Future Work

Deep learning continues to impress and astonish the world with its performance. However, the performance of a deep learning method depends on the training data provided to it. Due to the scarcity of "gold standard labeled" data, mostly training data are gathered by automatically labelling. Indeed, the automatically labeled data contain noise, which will affects the performance of deep neural networks. Most deep learning methods are not well equipped in handling noise labels. Noise in label data is harmful for the performance; therefore, the need for building noise-robust system has increased.

Deep Learning has shown some significant performance in the field of Natural Language Processing (NLP) due to its effective self-improvement and advancement learning. Part-of-speech tagging and Chunking are two core tasks in the NLP domain. The involvement of deep learning in these two tasks are immense. However, for most low resource languages deep learning methods are inefficient. Therefore, one way to handle such scenario is to generate automatic labels to increase the size of dataset and then training noise robust deep learning approach on it.

In our work, one possible approach that we used is called Noisy Labels Neural Network NLNN to handle noisy labels. It is a method to estimate the noise distribution in training labels and to generate an improved model of a neural network. NLNN uses Expectation and Minimization to estimate the hidden true labels (as the latent variable). There has been quite a few related work, but only NLNN can model the noise distribution. NLNN was implemented on MNIST dataset and TIMIT dataset. We explored NLNN for more complex NLP task. We re-implemented NLNN for POS tagging and Chunking task. NLNN successfully showed improvement in performance and indicates its strength to adapt for different challenging task.

Multi-tasking Learning is an approach which uses the related task information to improve generalization performance. MTL in recent times has a significant performance in various NLP task. To handle noise, we used the supervision signals from the related task.

NLNN is simple to implement, hence, we extended NLNN to MTL setup. The task involved POS tagging as low-level task and Chunking as high-level task. It was a challenging task, but NLNN performed well and gave the state-of-the-art performance for Chunking. These performances indicate that the NLNN can be used for several core NLP tasks and can also clean two or more tasks simultaneously.

In the next section, we would discuss some future works that which could be done to improve the performance of neural network on noise labels.

6.1 Future Work

An application of our work can be used to increase the amount of correct labeled data. One possible future direction could be to reuse the cleaned, labeled data to clean other noisy labeled data which is nothing but additive learning.

Figure 6.1, portrays the idea of additive learning, where the green color data signifies the clean data, and red part signifies noisy labeled data. In this method, we assumed that human data are clean labeled and artificial labeled data are partially noisy. In our experiments, we added 10%, 20%, 30 % and so on artificial data for the new experiment. Another approach could be to re-use the cleaned artificial labeled data in the new experiment. So, for 20% additional artificial data we could use the result of NLNN for 10% artificial labeled data and add 10% more artificial data. In this way, we are re-using the cleaner annotations which could improve the overall performance of NLNN on large data.

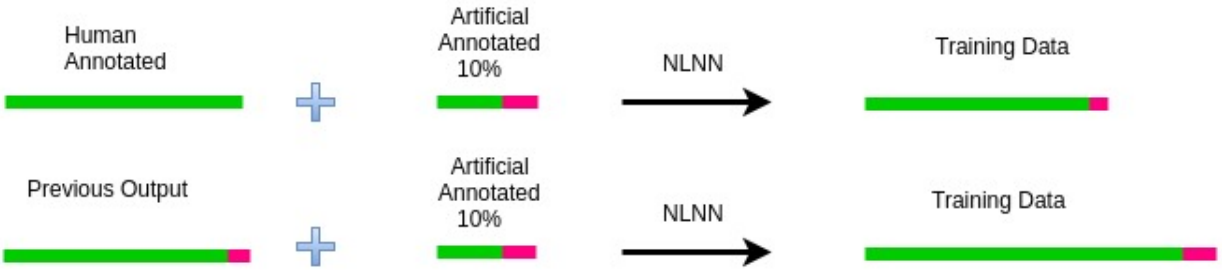

Figure 6.1: NLNN with Additive learning

Another possible future work will be to use the new version of NLNN with noise adaptation layer which is proposed in [17] and MTL to handle noise labels. During the experiment we have seen many error made by human annotators. In our experiments for MTL, we used two separate EM algorithms to estimate the noise distribution in two separate task. Perhaps, one can extend this idea and use EM algorithm to estimate the noise distribution in the human annotated data to improve the overall performance.

To summarize, our experiments showed the noise-robustness of NLNN on multiple NLP tasks simultaneously. NLNN can efficiently handle noise produced by artificial taggers. As the research in the field of neural network explores in several new directions, clean labeled data will always be of utmost important. Deep Learning approaches requires further concentration on developing noise-robust methods. Improved techniques might handle noisy labels efficiently, which will considerably be a large step in the field of machine learning.

Acronyms

Bi-LSTM Bidirectional Long Short-Term Memory. 28, 34, 36, 38–41, 47, 50, 51

EM Expectation and Minimization. 24, 43, 50, 52

ML Machine Learning. 1, 2

MTL Multi-tasking Learning. 3, 4, 30, 31, 44, 45, 50, 53

NANT North American News Text. 33, 38

NLNN Noisy Labels Neural Network. 3, 26, 30, 31, 34, 38, 40, 41, 44–48, 50–52

NLP Natural language processing. 1

NLU natural language understanding. 40

OOV out of vocabulary. 50

POS Part-of-speech. 2, 3, 27, 51

PTB Penn Treebank. 34, 40

STL Single-tasking Learning. 44

Bibliography

[1] Søgaard, Anders, and Yoav Goldberg. "Deep multi-task learning with low level tasks supervised at lower layers." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Vol. 2. 2016.

[2] Alonso, Héctor Martínez, and Barbara Plank. "When is multitask learning effective? Semantic sequence prediction under varying data conditions."

[3] Baxter, J. (1997). A Bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning, 28, 7–39.

[4] Phoneme duration and frequency profile Arık, S. Ö., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., ... Shoeybi, M. (2017). Deep Voice: Real-time Neural Text-to-Speech. In ICML 2017.

[5] Bengio, Y. (2009). Learning deep architectures for AI. Found. Trends Mach. Learn., 2(1), 1–127

[6] Deep Learning Use Cases - Data Science Pop-up Seattle 1. #datapopupseattle UNSTRUCTURED Data Science POP-UP in Seattle #datapopupseattle D 2. DEEP LEARNING USE CASES SKYMIND - 2015 *https://www.slideshare.net/dominodatalab/data-science-popup-seattle-deep-learning-use-cases*

[7] Bekker, Alan Joseph, and Jacob Goldberger. "Training deep neural-networks based on unreliable labels." Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE, 2016.

[8] Bingel, Joachim, and Anders Søgaard. "Identifying beneficial task relations for multi-task learning in deep neural networks." arXiv preprint arXiv:1702.08303 (2017).

[9] Eric Brill. 1992. A simple rule-based part of speech tagger. In Proceedings of the third conference on Applied natural language processing (ANLC '92). Association for Computational Linguistics, Stroudsburg, PA, USA, 152-155.

[10] Caruana, R. "Multitask learning: A knowledge-based source of inductive bias." Proceedings of the Tenth International Conference on Machine Learning. 1993.Caruana, R. (1998). Multitask Learning. Autonomous Agents and Multi-Agent Systems, 27(1), 95–133.

[11] Choi, E., Hewlett, D., Uszkoreit, J., Lacoste, A., & Berant, J. (2017). Coarse-to-Fine Question Answering for Long Documents. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (pp. 209–220).

[12] Dickinson, Markus, and W. Detmar Meurers. "Detecting errors in part-of-speech annotation." Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003.

[13] D. Nettleton, A. Orriols-Puig, & Fornells, A. (2010) A study of the effect of different types of noise on the precision of supervised learning techniques. Artificial intelligence review.

[14] Duong, L., Cohn, T., Bird, S., & Cook, P. (2015). Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), 845–850.

[15] Dong, D., Wu, H., He, W., Yu, D., & Wang, H. (2015). Multi-Task Learning for Multiple Language Translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (pp. 1723–1732).

[16] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani,and Eduard Hovy. 2013. Learning whom to trust with MACE. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Atlanta, Georgia, USA, NAACL-HLT'13, pages 1120–1130.

[17] Goldberger, Jacob, and Ehud Ben-Reuven. "Training deep neural-networks using a noise adaptation layer." (2016).

[18] Johnson, M., Schuster, M., Le, Q. V, Krikun, M., Wu, Y., Chen, Z., ... Dean, J. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation

[19] Jón Daníelsson, Financial Risk Forecasting:The Theory and Practice of Forecasting Market Risk, with Implementation in R and Matlab *http://onlinelibrary.wiley.com/doi/10.1002/9781119205869.app4/pdf*

[20] Larsen, J., Nonboe, L., Hintz-Madsen, M., & Hansen, L. K. (1998). Design of robust neural network classifiers. In IEEE International Con- ference on Acoustics, Speech and Signal Processing, 2, pp. 1205-1208.

[21] Lee, D. H. (2013). Pseudo-label: The simple and efficient semi- supervised learning method for deep neural networks. In Workshop on Challenges in Representation Learning, ICML, 3, p. 2.

[22] Liu, X., Gao, J., He, X., Deng, L., Duh, K., & Wang, Y.-Y. (2015). Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. NAACL-2015, 912–921

[23] Luong, Minh-Thang, et al. "Multi-task sequence to sequence learning." arXiv preprint arXiv:1511.06114 (2015).

[24] Megyesi, Beáta. "Brill's rule-based PoS tagger."

[25] Mnih, Volodymyr, and Geoffrey E. Hinton. "Learning to label aerial images from noisy data." Proceedings of the 29th International Conference on Machine Learning (ICML-12). 2012.

[26] N. Natarajan, I. Dhillon, P. Ravikumar, and A. Tewari. Learning with noisy labels. In Advances in Neural Information Processing Systems (NIPS), 2013.

[27] Nitin Hardeniya, Jacob Perkins, Deepti Chopra, Nisheeth Joshi, Iti Mathur."Natural Language Processing: Python and NLTK"

[28] Plank, Barbara, Anders Søgaard, and Yoav Goldberg. "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss." arXiv preprint arXiv:1604.05529 (2016).

[29] Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., & Pande, V. (2015). Massively Multitask Networks for Drug Discovery.

[30] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. Natural Language Processing (Almost) from Scratch, Journal of Machine Learning Research (JMLR), 2011.

[31] R. Collobert. Deep Learning for Efficient Discriminative Parsing, in International Conference on Artificial Intelligence and Statistics (AISTATS), 2011.

[32] Rich caruana, "Multitask Learning", caruana@cs.cmu.edu, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 Ed

[33] Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., & Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrap- ping. arXiv preprint arXiv:1412.6596.

[34] Rehbein, Ines, and Josef Ruppenhofer. "Detecting annotation noise in automatically labelled data." Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. 2017.

[35] Rolnick, David, et al. "Deep Learning is Robust to Massive Label Noise." arXiv preprint arXiv:1705.10694 (2017).

[36] Stubbs, Amber C. A methodology for using professional knowledge in corpus annotation. Brandeis University, 2013.

[37] Ruder, Sebastian. "An overview of multi-task learning in deep neural networks." arXiv preprint arXiv:1706.05098 (2017).

[38] Shen, Hong, and Anoop Sarkar. "Voting between multiple data representations for text chunking." Conference of the Canadian Society for Computational Studies of Intelligence. Springer, Berlin, Heidelberg, 2005.

[39] Steven Bird, Ewan Klein, and Edward Loper. "Natural Language Processing with Python"

[40] Sukhbaatar, S., & Fergus, R. (2014). Learning from noisy labels with deep neural networks. arXiv preprint arXiv:1406.2080, 2(3), 4.

[41] Toshniwal, S., Tang, H., Lu, L., & Livescu, K. (2017). Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition.

[42] Thanda, Abhinav, and Shankar M. Venkatesan. "Multi-task Learning Of Deep Neural Networks For Audio Visual Automatic Speech Recognition." arXiv preprint arXiv:1701.02477 (2017).

[43] Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., & Zhang, W. (2017). R$\hat{3}$: Reinforced Reader-Ranker for Open-Domain Question Answering.

[44] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In Advances in Neural Information Processing Systems (NIPS), 2005.

[45] Yang, Y., & Hospedales, T. M. (2017). Trace Norm Regularised Deep Multi-Task Learning. In Workshop track - ICLR 2017. Retrieved from http://arxiv.org/abs/1606.04038

[46] Zoph, B., & Knight, K. (2016). Multi-Source Neural Translation. NAACL, 30–34.

[47] X. Zhu and X. Wu. Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review, 22(3):177–210, 2004.