Training a Neural Network in a Low-Resource Setting on Automatically Annotated Noisy Data

Michael A. Hedderich[1,2] **Dietrich Klakow**[1]
[1]Spoken Language Systems (LSV)
[2]Saarbrücken Graduate School of Computer Science
Saarland Informatics Campus, Saarland University, Saarbrücken, Germany
{mhedderich,dietrich.klakow}@lsv.uni-saarland.de

Abstract

Manually labeled corpora are expensive to create and often not available for low-resource languages or domains. Automatic labeling approaches are an alternative way to obtain labeled data in a quicker and cheaper way. However, these labels often contain more errors which can deteriorate a classifier's performance when trained on this data. We propose a noise layer that is added to a neural network architecture. This allows modeling the noise and train on a combination of clean and noisy data. We show that in a low-resource NER task we can improve performance by up to 35% by using additional, noisy data and handling the noise.

1 Introduction

For training statistical models in a supervised way, labeled datasets are required. For many natural language processing tasks like part-of-speech tagging (POS) or named entity recognition (NER), every word in a corpus needs to be annotated. While the large effort of manual annotation is regularly done for English, for other languages this is often not the case. And even for English, the corpora are usually limited to certain domains like newspaper articles. For tasks in low-resource areas there tend to be no or only few labeled words available.

Distant supervision and automatic labeling approaches are an alternative to manually creating labels. These exploit the fact that frequently large amounts of unannotated texts do exist in the targeted domain, e.g. from web crawls. The labels are then assigned using techniques like transferring information from high-resource languages (Das and Petrov, 2011) or simple look-ups in knowledge bases or gazetteers (Dembowski et al., 2017). Once such an automatic labeling system is set up, the amount of text to annotate becomes nearly irrelevant, especially in comparison to manual annotation. Also, it is often rather easy to apply the system to different settings, e.g. by using a knowledge base in a different language.

However, while easily obtainable in large amounts, the automatically annotated data usually contains more errors than the manually annotated. When training a machine learning algorithm on such noisy training data, this can result in a low performance. Furthermore, the combination of noisy and clean training instances can perform even worse than just using clean data.

In this work, we present an approach to training a neural network with a combination of a small amount of clean data and a larger set of automatically annotated, noisy instances. We model the noise explicitly using a noise layer that is added to the network architecture. This allows us to directly optimize the network weights using standard techniques. After training, the noise layer is not needed anymore, removing any added complexity.

This technique is applicable to different classification scenarios and in this work, we apply it to an NER task. To obtain a non-synthetic, realistic source of noise, we use look-ups from gazetteers for automatically annotating the data. In the low-resource setting, we show the performance boost obtained from training with both clean and noisy instances and from handling the noise in the data. We also compare to another recent neural network noise-handling approach and we give some more insight into the impact of using additional noisy data and into the learned noise model.

2 Related Work

Existing work showed the importance of handling label noise. Zhu and Wu (2004) suggested that noise in labels tends to be more harmful than noise in features. Beigman and Beigman Klebanov (2009) showed that annotation noise in difficult instances can deteriorate the performance even on simple instances that would have been classified correctly in the absence of the hard cases.

Rehbein and Ruppenhofer (2017) presented a technique for detecting annotation noise in automatically labeled POS and NER tags in an active learning scheme. It requires, however, several sources of automatic annotations and human supervision. Similarly, Rocio et al. (2007) and Loftsson (2009) focused on detecting noisy instances in (semi-) automatically annotated POS corpora, leaving the correction to human annotators.

The model proposed by Bekker and Goldberger (2016) assumes that all clean labels pass through a noisy channel. One does only observe the noisy labels. The model of the noise channel, as well as the clean labels, are estimated using an EM algorithm. A neural network is then trained on the estimated labels. van den Berg (2016) applied this model to different tasks, obtaining small improvements on NER with automatically annotated data. A disadvantage of this approach is that the neural network needs to be retrained in every iteration of the EM algorithm, making the model difficult to scale to complex neural architectures.

Goldberger and Ben-Reuven (2017) transformed this model into an end-to-end trainable neural network by replacing the EM component with a noise adaptation layer. They experimented with simple image classification data and Dgani et al. (2018) applied it on the medical image domain. Both limit their approach to only using noisy data. Also, they just evaluate the effectiveness of their noise-handling method on simple synthetic noise (uniform and permutation). When applied to real-life scenarios, the noise might have a more complex structure.

In the image classification domain, several ideas have been proposed for estimating cleaned labels using a combination of clean and noisy labels. Fergus et al. (2009) employ a label propagation approach. Sukhbaatar et al. (2015) apply a noise model on top of a Convolutional Neural Network. Vahdat (2017) constructs an undirected graphical model to represent the relationship between clean and noisy labels. However, an additional source of auxiliary information is needed to infer clean from noisy labels. The approach presented by Veit et al. (2017) uses two components. A cleaning network learns to map noisy labels to clean ones. The second network is used to learn the actual image classification task from clean and cleaned labels. We compare our approach to this idea in the experiments.

3 Noise Layer

Given a clean dataset C consisting of feature and label tuples (x, y), we can construct a multi-label neural network softmax classifier

$$p(y = i|x; w) = \frac{\exp(u_i^T h(x))}{\sum_{j=1}^{k} \exp(u_j^T h(x))} \quad (1)$$

where k is the number of classes, h is a non-linear function or a more complex neural network and w are the network weights including the softmax weights u.

The noisy dataset N is a set of additional training instances. Following the approach of Goldberger and Ben-Reuven (2017), we assume that each originally clean (but unseen) label y went through a noise channel transforming it into the noisy label z. We only observe the noisy label, i.e. N consists of tuples (x, z).

The noise transformation from a clean label y with class i to a noisy label z with class j is modeled using a stochastic matrix

$$\theta(i, j) = p(z = j|y = i) = \frac{\exp(b_{ij})}{\sum_{l=1}^{k} \exp(b_{il})} \quad (2)$$

for $i, j \in \{1, ..., k\}$ and where b are learned weights. We call this the noise layer here. The probability for an observed, noisy label then becomes

$$p(z = j|x; w; \theta) = \sum_{i=1}^{k} p(z = j|y = i; \theta) p(y = i|x; w) \quad (3)$$

for $(x, z) \in N$.

In contrast to the work by Goldberger and Ben-Reuven (2017), we also have access to clean data C. From this, we create two models, as illustrated in Figure 1. The base model without noise layer is trained on C and the noise model with the noise layer is trained on N. Both models share the same

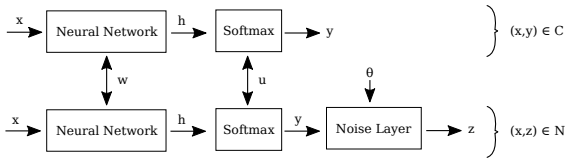

Figure 1: General architecture of the approach. Above is the base model trained on clean data C and predicting clean labels y. Below is the noise layer model trained on noisy label data N. The predicted labels y are transformed into the seen, noisy labels z using the noise layer.

network weights. The models are trained alternatingly, each for one epoch of its corresponding clean or noisy data. For prediction, the noise layer is removed and just the base model is used.

As stated by Goldberger and Ben-Reuven (2017), the initialization of θ is important. Since we have access to a small amount of clean data C, we use it for initializing the stochastic matrix. We assume that we can create noisy labels for the clean instances using the same process as for the noisy data N. We then initialize the weights of θ as

$$b_{ij} = \log\left(\frac{\sum_{t=1}^{|C|} 1_{\{y_t=i\}} 1_{\{z_t=j\}}}{\sum_{t=1}^{|C|} 1_{\{y_t=i\}}}\right) \quad (4)$$

where z_t is obtained by creating a noisy label for $(x_t, y_t) \in C$.

4 Dataset and Automatic Annotation

Named Entity Recognition (NER) is the task of assigning phrases in a text an entity label. In the sentence

> *Only France backed Fischler's proposal.*

the country *France* is of the entity class location and *Fischler* refers to a person. Creating training data for this task requires that each word in the text is labeled with its corresponding class. The effort to create a sufficiently large dataset might be too large for a low-resource language.

To tackle this problem, Dembowski et al. (2017) proposed to use external lists and gazetteers of entities to automatically label words in a training corpus. A list of person names can e.g be extracted from all of the entries appearing in Wikipedia's person category. Equipped with such lists for all entity classes, one can then label a text automatically. A word gets assigned a specific class if it appears in the corresponding entity list. A word or token that does not appear in any list gets assigned the null class "O". Additionally, simple heuristics help to resolve conflicts between lists and to remove some sources of errors. One might e.g. not label the day of the weeks as names, although *"Friday"* might be in the list of person names.

Class	Precision	Recall	F1
PER	48.09%	25.90%	33.67%
ORG	52.45%	10.02%	16.83%
LOC	56.76%	65.42%	60.78%
MISC	0.00%	0.00%	0.00%
Overall	53.31%	27.36%	36.16%

Table 1: Evaluation of the automatic labeling on the full English CoNLL-2003 training set (which we use as noisy dataset N).

For this work, we use the English CoNLL-2003 NER corpus (Tjong Kim Sang and De Meulder, 2003). The dataset is labeled with the classes person (PER), location (LOC), organization (ORG), miscellaneous name (MISC) and the null class (O). It consists of a training, a development and a test set. To obtain a low-resource setting, we randomly sample a subset of the training set as clean data C. In the experiments, we vary this size between ca. 400 and 20000 words. The rest of the labels are removed from the training set.

We then label the whole training set using the method by Dembowski et al. (2017) in the version with heuristics. This approach of automatically labeling words allows to quickly obtain large amounts of labeled text. However, both precision and recall tend to be lower than for manually labeled corpora (cf. Table 1). It should be noted that the MISC class is not covered with this technique which is an additional source of noise in the automatically annotated data. We use this as our noisy data N.

5 Model Architectures and Training

In this section, we present the different model architectures we evaluated in our experiments and we give details on the training procedure.

For each instance, the input x is a sequence of words with the target word in the middle surrounded by 3 words from the left and from the right of the original sentence, e.g. x = *"coun-*

tries other than Britain until the scientific" where *"Britain"* is the target word with label $y = \text{LOC}$. Sentence boundaries are padded. We encode the words using the 300-dimensional GloVe vectors trained on cased text from Common Crawl (Pennington et al., 2014).

The **base-model** uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with state size 300 to encode the input. Then a dense layer is applied with size 100 and ReLU activation (Glorot et al., 2011). Afterwards, the softmax layer is used for classification. This model is only trained on the clean data C.

The **noise-model** is built upon the base model and uses the noise layer architecture explained in section 3. First, the model is trained without noise layer for one epoch on the clean data. Then, we alternate between training with the noise layer on the noisy data and without the noise layer on the clean, each for one epoch. Instead of training on the full noisy corpus, we use a subsample $\tilde{N}$, randomly picked in each epoch. This allows the model to see many different noisy samples while preventing the noise from being too dominant. In section 6.2, we evaluate this effect.

For the **base-model-with-noise** we use the same clean and noisy data but the noise layer is left out, using only the base model architecture without an explicit noise-handling technique.

To evaluate the importance of the initialization of the stochastic matrix θ, the **noise-model-with-identity-init** uses the same training approach and data as the *noise-model*. However, θ is initialized with the identity matrix instead of using formula 4.

The **noise-adaptation-model** uses the original model of Goldberger and Ben-Reuven (2017). It consists of the base model with the noise layer and is trained on the whole noisy dataset in each epoch. It does not use the clean data. For initializing θ, the base model is pretrained on the noisy data and its predictions are used as an approximation to the clean labels.

We also compare to the recent work by Veit et al. (2017). They train a noise cleaning component which learns to map from a noisy label and a feature representation to a clean label. These cleaned labels are then used for training of what we call the base model. The authors did not report specific layer sizes and their architecture is developed for an image classification task, which differs structurally from our NER dataset (e.g. their label vector is much sparser). We, therefore, adapt their concept to our setting. As feature representation, we use the output of the BiLSTM which is projected to a 30-dimensional space with a linear layer. This is concatenated with the noisy label and used as input to the noise cleaning component. It is passed through a dense layer with the same dimension as the label vector. The skip-connection and clipping are used as in their publication. We use the same training approach and data as with the *noise-model*, replacing the step where the noise layer is trained. Instead, in each epoch the noise cleaning component is trained on C and the corresponding noisy labels. The base model is then trained on a cleaned version of $\tilde{N}$ and C. We call this the **noise-cleaning-model**.

All models are trained using cross-entropy loss, except for the noise cleaning component of the *noise-cleaning-model* which is trained with the absolute error loss like in the original paper. All models are trained for 40 epochs and the weights of the best performing epoch are selected according to the F1 score on the development set. Adam (Kingma and Ba, 2015) is used for stochastic optimization.

6 Experiments and Evaluation

In this section, we report on our experiments and their results. The training on noisy data as well as the randomness in training neural networks in general lead to a certain amount of variance in the evaluation scores. Therefore, we repeat all experiments five times and report the average as well as the standard error. To obtain meaningful results, no noise is added to the test data.

6.1 Model Comparison

To simulate different degrees of low-resource settings, we trained the models on different amounts of clean data. We vary the size between 407 labeled words (0.2% of the CoNLL-2003 training data) and 20362 labeled words (10%) in six steps. Since the noisy labels are easy to obtain, we use the whole corpus N. The size of the random subsample $\tilde{N}$ in each epoch is set to the same size as the clean data.

The results of this experiment are given in Figure 2. There is a general trend that the larger the amount of clean data is, the lower the differences between the models are. It seems that once we

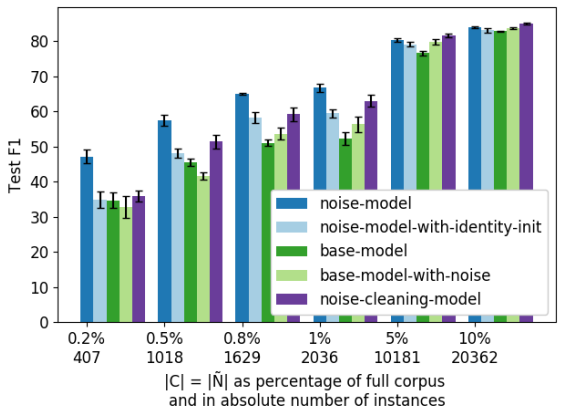

Figure 2: Evaluation results of the models. Experiments were run for different sizes of the clean data C and the per epoch randomly subsampled data $\tilde{N}$. The average F1 score on the test set is given over five runs. The error bars show two-times standard error in both directions.

have obtained enough clean training data, the additional noisy data cannot add much more information, even when cleaned. This is reminiscent of results from semi-supervised learning (e.g. in Nigam et al., 2006).

For the two settings with the lowest amount of data, the *base-model-with-noise* (which is trained on clean and noisy data without a noise channel) performs worst. For the four settings with more data, it is better than *base-model* (which is only trained on C). This could indicate that noisy labels do hurt the performance in low-resource settings. However, once a certain amount of clean training data is obtained, this is enough to cope with the noise to a certain degree and obtain improvements, even when the noise is not explicitly handled.

The models that do handle noise, outperform these baselines. When comparing *noise-model* and *noise-model-with-identity-init*, we see a large gap in performance. This shows the importance of a good initialization of the noise model θ in the low-resource setting.

The original *noise-adaptation-model* model by Goldberger and Ben-Reuven (2017) obtains an average F1 score of 38.8. This shows that a model purely trained on a large amount of automatically annotated data can be an alternative to a model trained on very few clean instances. However, the effect of cleaning noisy labels without access to any clean data seems limited, as the model cannot even reach the performance of either the *base-model* trained on 1018 instances nor our *noise-model* on the smaller set of 407 instances.

The *noise-model* outperforms the *cleaning-model* in the four lower-resource settings while the latter performs slightly better in the two scenarios with more data. With its access to the features in the noise cleaning component, the *cleaning-model* might be able to model more complex noise transformations. However, it does not seem to be able to leverage this capability in a low-resource setting. In the low-resource settings, our *noise-model* is able to handle the noise well and it gains over ten points in F1 score over not using a noise-handling mechanism or only training on clean data.

6.2 Amount of Noisy Data

In this experiment, we evaluate the effect of using different amounts of noisy data during each epoch, i.e. we vary the size of the subsampled, noisy data $\tilde{N}$. We experiment with the *noise-model* and fix the amount of clean data C to 2036 labeled words (1% of the CoNLL-2003 training data). We choose $|\tilde{N}|$ as multiples of $|C|$ using factors 0.5, 1, 2, 10, 20, 30 and 50.

The results are given in Figure 3. One can see a trend that increasing the size of $\tilde{N}$ results in an improvement in F1 score. This holds until factor 5. Afterwards, the performance degrades again. This might indicate that the noisy data becomes too dominant and the cleaning effect of the noise layer is not able to mitigate it.

6.3 Learned Weights

Since, for evaluation purposes, we do have access to the clean labels of the whole training set, we can compare the noise that is in the noisy data to what the noise layer learned. Table 1 shows the evaluation of the automatically annotated labels on the training data. Figure 4 shows the stochastic matrix θ that was learned in one run of training the *noise-model* with $|C| = |\tilde{N}| = 2036$ labeled words (1% of the CoNLL-2003 training data).

One can see that the learned weight matrix represents a reasonable model of the noise. For the classes PER, ORG and MISC, the recall is very low in the noisy data and therefore the corresponding weights in the first column of the matrix are high: Instances (or a certain percentage of the probability mass) which the base model correctly classifies as PER/ORG/MISC, are mapped to the class O because this is the corresponding noisy label (indicated by the low recall). For the LOC

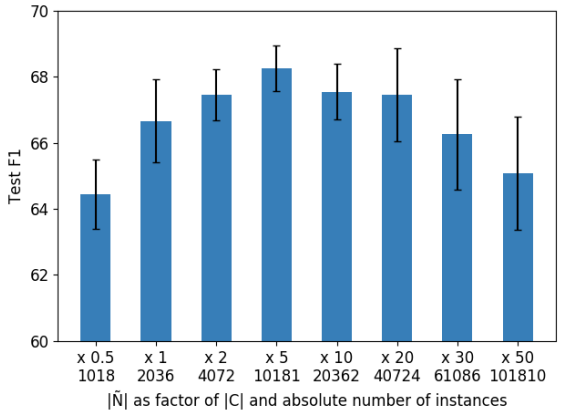

Figure 3: Evaluation results for varying the size of the per epoch randomly subsampled noisy data $\tilde{N}$. The *noise-model* was used and the amount of clean data C fixed to 2036 labeled words. The average F1 score on the test set is given over five runs. The error bars show two-times standard error in both directions.

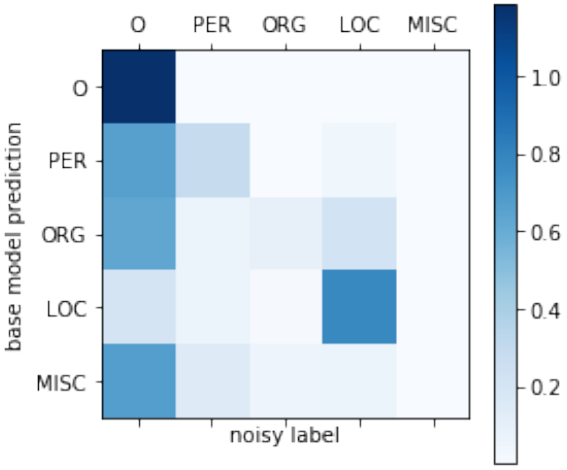

Figure 4: Representation of the noise transition weights θ learned in the noise layer. Each square is a value $\exp(\theta_{ij})$ where i is the vertical and j the horizontal index in the visualization.

class, the recall in the noisy labels is much higher and we see this reflected in the learned weights. The prominent weight is $\theta_{\text{LOC, LOC}}$, i.e. a prediction of the label LOC is mostly left unchanged because it tends to be correctly labeled in the noisy data.

7 Conclusions and Future Work

In this work, we presented a technique to train a neural network on a combination of clean and noisy annotations. We modeled the noise explicitly using a noise layer. We evaluated our approach on an NER task using real noise in the form of automatically annotated labels. We found that the probabilistic noise matrix learned is a useful model of the noise. In the low-resource setting where only a few manually annotated instances are available, we showed the improvements of up to 35% obtained from using additional, noisy data and handling the noise.

For future work, we want to experiment with different classification tasks and other sources of noisy data. We would also like to explore more complex noise models that are able to perform well both in low- and high-resource settings.

References

Eyal Beigman and Beata Beigman Klebanov. 2009. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

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

Esther Maria van den Berg. 2016. Noisy label neural network approach to named entity recognition. Master's thesis, Rijksuniversiteit Groningen, Universität des Saarlandes.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Julia Dembowski, Michael Wiegand, and Dietrich Klakow. 2017. Language independent named entity recognition using distant supervision. In *Proceedings of Language and Technology Conference (LTC)*.

Yair Dgani, Hayit Greenspan, and Jacob Goldberger. 2018. Training a neural network based on unreliable human annotation of medical images. In *IEEE International Symposium on Biomedical Imaging (ISBI)*.

Rob Fergus, Yair Weiss, and Antonio Torralba. 2009. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *Int. Conference on Learning Representations (ICLR)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Hrafn Loftsson. 2009. Correcting a pos-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*.

Kamal Nigam, Andrew McCallum, and Tom Mitchell. 2006. Semi-supervised text classification using em. *Semi-Supervised Learning*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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

Vitor Rocio, Joaquim Silva, and Gabriel Lopes. 2007. Detection of strange and wrong automatic part-of-speech tagging. In *Progress in Artificial Intelligence*.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir D. Bourdev, and Rob Fergus. 2015. Training convolutional networks with noisy labels. In *ICLR Workshop track*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*.

Arash Vahdat. 2017. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xingquan Zhu and Xindong Wu. 2004. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22.