EXPLORATION OF NOISE STRATEGIES IN SEMI-SUPERVISED NAMED ENTITY
CLASSIFICATION

by

Pooja Lakshmi Narayan

Copyright © Pooja Lakshmi Narayan 2019

A Thesis Submitted to the Faculty of the

DEPARTMENT OF COMPUTER SCIENCE

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2019

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Master's Committee, we certify that we have read the thesis prepared by Pooja Lakshmi Narayan, titled Exploration of Noise Strategies in Semi-Supervised Named Entity Classification and recommend that it be accepted as fulfilling the dissertation requirement for the Master's Degree.

Mihai Surdeanu

Date: 8/29/2019

Joshua Levine

Date: 8/29/2019

Carlos Scheidegger

Date: 2019-08-29

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to the Graduate College.

I hereby certify that I have read this thesis prepared under my direction and recommend that it be accepted as fulfilling the Master's requirement.

Mihai Surdeanu
Master's Thesis Committee Chair
Computer Science

Date: 8/29/2019

Acknowledgements

I would like to thank my friends and family for constant support. My son for being there and troubling less. My mother and husband for believing in me when I was not sure about myself. Ajay introduced me to the world of research and rekindled my zeal to learn and experiment with new ideas. With persistent support from both my parents and my brother I believe it was possible. Last but not the least, Thanks to Prof.Mihai for helping me with the admission process without which I probably would not have joined the masters program itself and of course for being an amazing advisor.

Contents

List of Figures	5
List of Tables	6
Abstract	7
Chapter 1 Introduction	8
Chapter 2 Background	10
Supervised Machine Learning	10
Semi-supervised Learning	12
Shortcomings of Bootstrapping	13
Semi-supervised Deep Learning	14
Mean-teacher Framework	14
Mean Teacher vs. Other Semi-supervised Learning Algorithms	16
Chapter 3 Exploration of Noise Strategies for the NEC Task	17
Named Entity Classification	17
Noise Strategies	18
Chapter 4 Experiments	21
Task and Datasets	21
Experimental Settings	21
Results	22
Discussion	24
Chapter 5 Conclusion and Future Work	26
References	27

List of Figures

2.1	The <i>mean teacher</i> framework. $cost = (classification_cost) + \lambda(consistency_cost)$. Unlabeled examples have only consistency cost. Backpropagation only through <i>student model</i> , <i>teacher model</i> parameters are averaged.	14
3.1	Mean Teacher framework for the named entity classification task (left). E_{wi} are words in the entity mention, W_i are words in the context with entity mention replaced by $\langle E \rangle$ token. $cost = (classification_cost) + \lambda(consistency_cost)$. Unlabeled examples have only consistency cost. Backprop only through student model, teacher model parameters are averaged.	18
3.2	The architecture of the student or teacher model (right). Noise can be added to parts in shown boldface. They are (1) input noise (2) statistical noise (3) network dropout. $predictions = softmax(output_layer)$	18
4.1	CoNLL dataset histogram of the labels	22
4.2	OntoNotes dataset histogram of the labels	22
4.3	Performance upon combining noise strategies, CoNLL. Best performance: <i>network dropout + 5W word-dropout</i> - 70.57% (CoNLL)	23
4.4	Performance upon combining noise strategies Ontonotes. Best performance: <i>network dropout + 3W synonym-replace</i> - 72.78% (Ontonotes)	24

List of Tables

4.1	Overall accuracies comparing all noise strategies on CoNLL and Ontonotes datasets. <i>No noise</i> is the baseline. $X \ W \Rightarrow$ X words perturbed by noise. Accuracy is % of correctly classified datapoints. $(\pm y) \Rightarrow$ variance of 5 runs.	23
4.2	Tuning Gaussian noise - #words & stdev	24
4.3	Performance upon combining noise strategies on CoNLL and Ontonotes with accuracy and stdev	25

Abstract

Noise is inherent in real world datasets and modeling noise is critical during training, as it is effective in regularization. Recently, novel semi-supervised deep learning techniques have demonstrated tremendous potential when learning with very limited labeled training data in image processing tasks. A critical aspect of these semi-supervised learning techniques is augmenting the input or the network with noise to be able to learn robust models. While modeling noise is relatively straightforward in continuous domains such as image classification, it is not immediately apparent how noise can be modeled in discrete domains such as language. Our work aims to address this gap by exploring different noise strategies for the semi-supervised named entity classification task, including statistical methods such as adding Gaussian noise to input embeddings, and linguistically-inspired ones such as dropping words and replacing words with their synonyms. We compare their performance on two benchmark datasets (OntoNotes and CoNLL) for named entity classification. Our results indicate that noise strategies that are linguistically informed perform at least as well as statistical approaches, while being simpler and requiring minimal tuning.

1 Introduction

Modeling noise is a fundamental aspect of machine learning systems. The real world where these systems are deployed are certainly exposed to noisy data. Furthermore, noise is used as an effective regularizer during the training of neural networks (*e.g.*, dropout (Srivastava et al., 2014)¹). Correct prediction in the presence of noisy input demonstrates robustness of learning systems. A simple analogy to illustrate this is, during image classification, the addition of limited random Gaussian noise to an image can be barely perceived by our visual system and does not drastically change the label a human assigns to an image (Raj, 2018)². With the emphasis on compliance and recent advances in adversarial techniques, modeling noise has assumed renewed importance (Goodfellow et al., 2014)³.

Supervised learning, the most widely used paradigm in machine learning requires high quality labeled training data which is expensive and requires manual labor. Semi-supervised learning alleviates this problem by being less dependent on labeled training data. However, classical iterative semi-supervised algorithms have the problem of semantic drift, in which a small error introduced during initial phase of the training can drift the learning in a wrong direction (elaborated further in Chapter 2). Hence in this work we use one-shot learning algorithms (explained further in Chapter 2)

Noise is an important factor in recent state-of-the-art semi-supervised learning systems for image classification (Tarvainen and Valpola, 2017; Rasmus et al., 2015; Miyato et al., 2018b)⁴. In image processing, modeling random noise is relatively straightforward as it is a continuous domain. For instance, adding a small amount of random Gaussian jitter can be considered as noisy input. So are other image transformations such as translation, rotation, removing color and so on. However, a discrete domain such as language is not easily flexible to noise augmentation. While one can certainly add random Gaussian noise to embeddings of words (continuous vector representation such as *word2vec* (Mikolov et al., 2013)⁵ rather than one-hot encoding), the intuition behind such perturbation is not apparent. Algorithms which require explicit modeling of noise require careful thinking in the language domain and is challenging (Clark et al., 2018; Nagesh and Surdeanu, 2018a)⁶.

To the best of our knowledge, previous work in the area of modeling noise

¹ Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**, pp. 1929–1958

² Raj, B. (2018). Data Augmentation - How to use Deep Learning when you have Limited Data - Part 2. <https://bit.ly/2IvKw11>. Accessed: 2018-12-10

³ Goodfellow, I. J., J. Shlens, and C. Szegedy (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*

⁴ Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, **abs/1703.01780**; Rasmus, A., H. Valpola, M. Honkala, M. Berglund, and T. Raiko (2015). Semi-Supervised Learning with Ladder Network. *CoRR*, **abs/1507.02672**; and Miyato, T., S. Maeda, S. Ishii, and M. Koyama (2018b). Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1. ISSN 0162-8828. DOI: 10.1109/TPAMI.2018.2858821

⁵ Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.) *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc

⁶ Clark, K., T. Luong, C. D. Manning, and Q. V. Le (2018). Semi-Supervised Sequence Modeling with Cross-View Training; and Nagesh, A. and M. Surdeanu (2018a). An Exploration of Three Lightly-supervised

in natural language processing (NLP) applications has been limited. Clark et al. (2018)⁷ acknowledge the difficulty of modeling noise for language and incorporate a simple word dropout in their experiments. So does the work by Nagesh and Surdeanu (2018a)⁸. Nagesh and Surdeanu (2018b)⁹ add a standard Gaussian perturbation with a fixed variance to the pre-trained word vectors to simulate noise. Belinkov and Bisk (2017)¹⁰ is perhaps one of the most comprehensive works that explore various noise strategies with a different end goal in mind. Their work explores the degree of robustness of various neural network approaches to different types of noise on a machine translation task.

In this report, we discuss several noise strategies for the semi-supervised named entity classification task. Some of these, such as word-dropout and synonym-replace, are linguistic and are discrete in nature while others such as Gaussian perturbation to word embeddings are statistical. We show that linguistic noise, while being simple, perform as well as statistical noise. A combination of linguistic and network dropout provides the best performance. The various noise strategies are discussed in detail in Chapter 3. The next chapter gives more background on supervised machine learning, semi-supervised learning and semi-supervised deep learning which is not iterative (referred to as one-shot learning algorithms). In the Chapter 4 we discuss the experimental settings and results of the various noise strategies. We show that a hybrid of linguistically inspired noise and network dropout in the neural network performs well empirically and gives the best performance. Linguistic and network noise strategies provide complementary information, so the best approach learns to capture information from both these type of signals. This is followed by Chapter 5 which summarizes our contributions and some ideas for future work.

⁷ Clark, K., T. Luong, C. D. Manning, and Q. V. Le (2018). Semi-Supervised Sequence Modeling with Cross-View Training

⁸ Nagesh, A. and M. Surdeanu (2018a). An Exploration of Three Lightly-supervised Representation Learning Approaches for Named Entity Classification. In *COLING*

⁹ Nagesh, A. and M. Surdeanu (2018b). Keep your bearings: Lightly-supervised Information Extraction with Ladder Networks that avoids Semantic Drift. In *NAACL HLT 2018*

¹⁰ Belinkov, Y. and Y. Bisk (2017). Synthetic and Natural Noise Both Break Neural Machine Translation. *CoRR*, **abs/1711.02173**

2 Background

Our focus in this research work has been in the semi-supervised setting where the availability of labeled data is limited. But we would like to elaborate on supervised learning, its limitations and why semi-supervised learning is an interesting and important problem to solve. In this chapter we also elaborate on a deep learning framework called Mean Teacher which is the architecture we work with and also why we chose to work with it.

Supervised Machine Learning

Machine learning is a suite of algorithms that have the ability to learn from data. According to a widely accepted definition, "*learning is a computer program that learns from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .*"(Mitchell, 1997)"¹.

An example of a task T is classifying an email as spam or regular mail which is basically a binary classification task. The performance measure P on this task could be the accuracy of classification *i.e.*, among the predictions by the spam classifier program, how many are actually correctly classified (either as spam or regular mail).

The crucial aspect of any machine learning system is how to provide it with experience E . Supervised machine learning provides experience as a set of labeled examples that can be used to *train* the system to be able to learn with experience. In the case of the spam classifier system, it is a bunch of emails which are labeled by hand as spam or not spam mails. This is typically provided by human annotators, by looking at the content of emails. The learning algorithm uses the annotated data as training data and improves its predictions by iterating over this dataset, typically optimizing an objective or cost function which updates the parameters of the algorithm by encouraging correct classifications and penalizing incorrect ones thereby *learning* the optimal values of the parameters until acceptable performance is achieved.

Supervised machine learning is one of the most widely used paradigms in machine learning literature. Although supervised learning achieves very high performance on several learning tasks, it requires labeled data for training.

A typical machine learning pipeline consists of splitting the labeled dataset

¹ Mitchell, T. M. (1997). *Machine learning*. McGraw Hill series in computer science. McGraw-Hill. ISBN 978-0-07-042807-2

into three parts.

1. A *train dataset*, used to train the parameters of a machine learning program (also called model or parameters)
2. A *development dataset* (or *validation dataset*), used to tune the parameters of the tuned model, typically used to improve the generalizability of a model and avoid overfitting the model to the training dataset
3. A *test dataset* (or *evaluation dataset*), used to determine the performance of the algorithm on this dataset. The test set is typically treated as *blind* set, *i.e.* the training algorithm is not allowed to look at this dataset, to simulate a real world setting.

According to recommended practices, 80% of the labeled dataset is used for training, 10% as development dataset and the remaining 10% as test dataset (Moindrot and Genthial, 2018) ². When to stop the *learning* process to avoid over-fitting the training data can be analysed by plotting the train and test learning curves. The point at which train error is decreasing and test error is increasing is when it is said to over-fit the parameter values of the train data instead of generalizing to the larger problem (Goodfellow et al., 2016) ³. One of the common strategies used to attain better model is shuffling the training data each time (called epoch) and then running the training process again helps train better model parameters. Choosing how many epochs to run is a hyperparameter and is determined using these curves.

To achieve good performance, we need a large labeled dataset for training the parameters of a model in a supervised machine learning system. This dependence on the creation of labeled training data could be a severe limitation in widely deploying learning algorithms for the following reasons:

1. Creation of labeled dataset is a costly and laborious task. It involves several subtasks such as setting up the annotation task carefully with clear guidelines to the annotators, tool for annotation, employing human annotators, redundancy in annotation of data and inter-annotator agreement to make the data robust and so on.
2. For a new domain or language, where resources are hard to come by, this is a very limiting constraint for employing learning algorithms on such tasks. (*e.g.* spam classification on facebook posts (new domain) or mails in a non-English language (new language)).

Among several paradigms of Machine Learning, there exist some which try to alleviate this over-dependence on labeled training dataset of supervised machine learning. For instance, unsupervised learning or clustering, which does not require any labeled datasets. These have their advantages and can be used for visualizing datasets among other applications. However, due to the absence of labels, they are not precise in many learning tasks. Semi-supervised learning is a paradigm which bridges this gap and attempts to

² Moindrot, O. and G. Genthial (2018). Splitting into train, dev and test sets: Best practices to split your dataset into train, dev and test sets. <https://cs230-stanford.github.io/train-dev-test-split.html>. Accessed: 2018-04-03

³ Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. The MIT Press. ISBN 0262035618, 9780262035613

bring the best of both worlds together by avoiding the over-dependence of labeled data in completely supervised learning and the imprecise nature of purely unsupervised learning techniques. In this paradigm, there exists a labeled dataset which is smaller in comparison with supervised learning and a large unlabeled dataset which is used in addition with the small labeled dataset to learn a model with good performance. Since the labeled dataset during training is small, it does not require very large annotated datasets. Since there is a minimal amount of supervision, the training task is not imprecise as in the unsupervised learning setting.

We explain this paradigm in detail in the next section.

Semi-supervised Learning

According to the Encyclopedia of Machine Learning, semi-supervised learning is defined as follows, "semi-supervised learning uses both labeled and unlabeled data to perform an otherwise supervised learning or unsupervised learning task." (Sammut and Webb, 2011)⁴. Generally, the number of unlabeled examples are much larger than the number of labeled examples.

The problem setup in semi-supervised learning is as follows: Suppose there are N labeled examples

$$(x_1, y_1^*), (x_2, y_2^*) \dots (x_N, y_N^*)$$

and M unlabeled examples

$$x_{N+1}, x_{N+2} \dots x_{N+M}$$

where $M \gg N$. The goal is to learn a function $P(y|x)$ by using both the labeled and unlabeled examples.

Semi-supervised learning (SSL) is one of the cornerstones in machine learning (ML). This is especially true in the case of natural language processing (NLP), as obtaining labeled training data is a costly and tedious process for most of the data-hungry deep learning models.

Semi-supervised learning has been an active area of research for decades and a rich body of work exists in this area, comprising of several different techniques and algorithms. A survey of the various semi-supervised learning algorithms can be found at Zhu (2005)⁵ and Abney (2007)⁶.

Some of the commonly employed techniques are self-training (Yarowsky, 1995)⁷, co-training (Yangarber, 2003; Riloff, 1996)⁸, label propagation (Zhu and Ghahramani, 2002)⁹. Bootstrapping¹⁰ is a popular self-training algorithm widely used in several NLP tasks. In this approach, the set of labeled examples, also called seed examples, are given as input to the algorithm. Seed examples are carefully selected to be a representative set of labels that aid the algorithm to make predictions on unlabeled data. That is, one may not choose all or most seeds from one particular class of labels.

⁴ Sammut, C. and G. I. Webb (2011). *Encyclopedia of Machine Learning*. Springer Publishing Company, Incorporated, 1st edition. ISBN 0387307680, 9780387307688

⁵ Zhu, X. (2005). Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison

⁶ Abney, S. (2007). *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 1st edition. ISBN 1584885599, 9781584885597

⁷ Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*

⁸ Yangarber, R. (2003). Counter-training in discovery of semantic patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*; and Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pp. 1044–1049

⁹ Zhu, X. and Ghahramani, Z. (2002).

Most commonly, it is proportional to the distribution of labels in each class. But since in semi-supervised learning we are not aware of the actual label distribution, we just take equal number of seeds from each label type.

Some examples from literature: named entity classification (Collins and Singer, 1999)¹¹, relation extraction (Carlson et al., 2010a,b)¹², domain specific terminology extraction (Gupta and Manning, 2014)¹³, (Gupta and Manning, 2015)¹⁴ and so on.

The algorithm starts with seed examples and iterates through the following two steps:

1. Find the "best" set of patterns that are indicative of seed examples.
2. Find other examples in the unlabeled dataset that co-occur with the selected patterns and select the "top" examples (depending on their confidence scores) to be added to the seed examples, thus expanding the seed example set.

The algorithm stops after a fixed number of iterations or when it cannot find new examples to be added to the seed collection.

An example of bootstrapping in action is the task of named entity classification (NEC). The NEC task is that of labeling a given span of text, say, *Barack Obama*, by one of the many entity types, such as PER, ORG, LOC or MISC which stands for person, organization, location or miscellaneous (in this example it is PER). The algorithm is given seed entities. This could be common or famous names of PER, ORG, LOC and MISC. The algorithm iterates several times repeating the two steps mentioned above. In the first step, it learns patterns that co-occur with the seed entities and selects some representative patterns such as *@ENTITY said*, *@ENTITY, president of* and so on. In the second step, it finds non-seed entities in text that co-occur with these patterns and selects the most promising set and expands the seed entities.

Shortcomings of Bootstrapping

Bootstrapping generally works well in several applications (Carlson et al., 2010a)¹⁵. However, among other things it has one major shortcoming known as *semantic drift* (McIntosh, 2010)¹⁶. It is illustrated with an example: Suppose we are learning names of women using the bootstrapping algorithm defined above and we start with a seed name such as *Lily*. The algorithm might proceed to find patterns which are names of flowers in the subsequent iterations of the algorithm. The algorithm thereby deviates from the original semantics of the class being bootstrapped and adds impurities to the seed examples. It can be shown that semantic drift is unavoidable in bootstrapping due to the iterative nature of the algorithm (Komachi et al., 2008)¹⁷.

¹¹ Collins, M. and Y. Singer (1999). Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*

¹² Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell (2010a). Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*; and Carlson, A., J. Betteridge, R. C. Wang, E. R. Hruschka Jr., and T. M. Mitchell (2010b). Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 101–110. ACM

¹³ Gupta, S. and C. D. Manning (2014). Improved Pattern Learning for Bootstrapped Entity Extraction. In *CoNLL*, pp. 98–108

¹⁴ Gupta, S. and C. D. Manning (2015). Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*

¹⁵ Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell (2010a). Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*

¹⁶ McIntosh, T. (2010). Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 356–365. Association for Computational Linguistics

¹⁷ Komachi, M., T. Kudo, M. Shimbo, and Y. Matsumoto (2008). Graph-based Analysis of Semantic Drift in Espresso-like Bootstrapping Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '08*, pp. 1011–1020. Association for Computational Linguistics, Stroudsburg, PA, USA

Semi-supervised Deep Learning

There has been a flurry of recent work in SSL in the image processing community (Tarvainen and Valpola, 2017; Rasmus et al., 2015)¹⁸. Some of these recent works have achieved impressive performance on hard perceptive tasks.

However, adapting these works to NLP is not a straight forward exercise. As stated earlier, many of these approaches require noise (along with an optional input augmentation step such as rotation) to change the percept slightly, to achieve robust performance. However, augmenting data with noise for NLP tasks is not very clear, as the input domain consists of discrete tokens rather than continuous inputs such as images. We discuss the modeling of noise specifically in the framework which is best suited for natural language processing in the following section and in more detail in Chapter 3.

In the current work, we explore the *mean-teacher* (MT) framework (Tarvainen and Valpola, 2017)¹⁹ as the basic algorithm for the semi-supervised named entity classification task. In the following section, we explain the mean teacher framework and situate it in the broader class of semi-supervised deep learning algorithms in recent literature.

Mean-teacher Framework

The architecture of the *mean teacher* (MT) framework is depicted in Figure 2.1. MT framework belongs to the general class of teacher-student networks that learns in the semi-supervised setting *i.e.*, limited supervision and a large amount of unlabeled data and is illustrated in Figure 2.1.

It consists of two models, termed *student* and *teacher* which are structurally identical but differ in the way their parameters are updated. While the student is updated using regular back-propagation, the parameters of the

¹⁸ Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, **abs/1703.01780**; and Rasmus, A., H. Valpola, M. Honkala, M. Berglund, and T. Raiko (2015). Semi-Supervised Learning with Ladder Network. *CoRR*, **abs/1507.02672**

¹⁹ Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, **abs/1703.01780**

Figure 2.1: The *mean teacher* framework. $cost = (classification_cost) + \lambda(consistency_cost)$. Unlabeled examples have only consistency cost. Backpropagation only through *student model*, *teacher model* parameters are averaged.

teacher are a weighted average of the student parameters across different epochs.

The intuition of the mean-teacher framework for the semi-supervised learning problem is as follows. The student on its own will semantically drift like any SSL algorithm which is iterative in nature. To keep on track, the MT algorithm uses a “scaffolding” model, the mean teacher. This teacher is likely to be more stable, less influenced by irregularities in the data and mitigates semantic drift because it averages previous students. As it has been shown in the literature (Collins, 2002; Tarvainen and Valpola, 2017)²⁰, such an averaging brings more stability. Thus, the MT algorithm encourages the student to stay close to the teacher through a consistency loss that penalizes large differences between the two.

Each epoch has multiple mini batches during its training step. And each mini-batch consists of some labeled examples and the rest is unlabeled. Since the same set of labeled examples is sampled randomly in each mini-batch there is some sort of over sampling from the set of labeled examples. One of the downsides of oversampling is that the training process overfits on those specific samples. To mitigate this we use noise which is like shaking up the same data point a bit so that its representation is similar but we do not overfit with that datapoint itself. It helps in some way of regularizing the model thus increasing its robustness and helps in yielding better performance than the no-noise model.

Further, the cost function is a linear combination of supervision cost (from the limited number amount of supervision) and consistency cost (agreement between the representation from the teacher and student models measured as the L^2 norm difference between them). The motivation of using consistency in the cost function and averaging the parameters in the teacher is to reduce confirmation bias in the teacher when its own predictions are used as pseudo-labels during the training process (akin to averaged perceptron). This provides a strong proxy for the student to rely on, in the absence of labeled training data.

The student learns from the small set of labeled examples and on the remaining (large) number of unlabeled examples it relies on the “judgement” of the teacher to make the right predictions. While there is the issue of confirmation bias with the student and teacher artificially agreeing with each other to reduce consistency cost, this can be mitigated in several ways. One of the most important ways to reduce the confirmation bias is the incorporation of noise in the models (both input noise as well as noise in the architecture).

The intuition for adding noise can be explained through the example of image processing. Suppose the task is that of image classification, a slight perturbation of the image (adding a little bit of random noise in the image) should not drastically alter the label given to that image. This is broadly along the lines of human visual system. Robustness to noise will allow the system to learn the important latent features in the image. As depicted in

²⁰ Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pp. 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA. DOI: 10.3115/1118693.1118694; and Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, [abs/1703.01780](https://arxiv.org/abs/1703.01780)

Figure 2.1, noise becomes an important component in the mean teacher framework. We explain this in detail in Chapter 3.

Mean Teacher vs. Other Semi-supervised Learning Algorithms

As mentioned in Chapter 1 and in the beginning of the current Chapter, there are several paradigms in the broad area of semi-supervised learning. In the current section, we aim to situate the mean teacher framework in this area and explain the reasoning behind our selection of the mean-teacher framework.

In a previous work Nagesh and Surdeanu (2018a) ²¹ evaluate three different semi-supervised learning paradigms, namely, bootstrapping-based approaches (Gupta and Manning, 2015) ²², ladder networks (Rasmus et al., 2015) ²³ and mean-teacher (Tarvainen and Valpola, 2017) ²⁴ for the semi-supervised named entity classification (NEC) task. The outcome of this study is that, semi-supervised learning strategies could be categorized into those that are iterative in nature and those that are one-shot learning algorithms. Being iterative, the bootstrapping based techniques are compelled to choose pseudo-label(s) in the intermediate stages of this procedure, thereby making hard decisions about the labels. However, one-shot learning techniques (such as ladder networks and mean teacher) have a soft assignment of pseudo-labels during the intermediate training steps and are not compelled to make such sub-optimal decisions. Further it was shown to mitigate semantic drift. As a result, the mean-teacher (MT) approach was a top performing system and the most flexible framework among the three. However, the exploration of noise was limited in the previous study and hence is the focus of the current report.

In the wider area of semi-supervised deep learning, there are several other techniques such as temporal ensembling, pi-networks (Laine and Aila, 2016) ²⁵ and virtual adversarial training (Miyato et al., 2018a) ²⁶. All of these can be studied as teacher-student models (Tarvainen and Valpola, 2017) ²⁷. The goal of each of these systems is to be able to learn a robust way to assign pseudo-labels to unlabeled examples without having confirmation bias. The way they achieve this goal is to have two competing/co-operating models (*teacher* and *student*). One can achieve this goal by making the student task harder by choosing to add noise that would confuse the student model (virtual adversarial training, for instance) or make the teacher task easier (by ensembling or averaging the weights / decisions of the various student models leading to the teacher model). Both are complementary approaches and the mean-teacher framework chooses the latter. The temporal ensembling only keeps around the last "K" students but mean-teacher is a mean (average) over many students with an exponential moving average to give weightage to the more recent ones.

In Chapter 3, we discuss the task of named entity classification and the noise strategies that we explored as part of this project.

²¹ Nagesh, A. and M. Surdeanu (2018a). An Exploration of Three Lightly-supervised Representation Learning Approaches for Named Entity Classification. In *COLING*

²² Gupta, S. and C. D. Manning (2015). Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*

²³ Rasmus, A., H. Valpola, M. Honkala, M. Berglund, and T. Raiko (2015). Semi-Supervised Learning with Ladder Network. *CoRR*, **abs/1507.02672**

²⁴ Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, **abs/1703.01780**

²⁵ Laine, S. and T. Aila (2016). Temporal Ensembling for Semi-Supervised Learning. *CoRR*, **abs/1610.02242**

²⁶ Miyato, T., S. ichi Maeda, M. Koyama, and S. Ishii (2018a). Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning. *IEEE transactions on pattern analysis and machine intelligence*

²⁷ Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, **abs/1703.01780**

3 *Exploration of Noise Strategies for the NEC Task*

A critical component in the algorithm is the addition of noise to the models. Noise can be added mainly in three key places to the model presented in the previous section as depicted in Figure 3.2.

Named Entity Classification

The specific model we employ for semi-supervised named entity classification (NEC) task along with a representative input data point is depicted in the Figures 3.1 and 3.2. The input consists of an entity mention and the sentence it appears in, as the context. The goal is to predict the label of the entity. In the semi-supervised setting only a few labeled data points are provided, the rest of the data is unlabeled.

Each word is represented as a high dimensional vector and they are grouped together by distributional similarity. This is a standard algorithm called word2vec (Mikolov et al., 2013)¹ and these are mostly trained offline on large documents. These pre-trained word embeddings serve as a starting point for most NLP tasks since its trained in such a way that similar words or synonyms are closer in the vector space and say antonyms are farther apart. Since language has rich sequence, the semantics of a word would change depending on the words before it and the words that follow. Hence we use a *recurrent neural network* (RNN) to capture this sequence in the sentence. An RNN consumes tokens in a sequence and computes a latent representation of the sequence seen so far, essentially encoding various information seen in the sequence such as syntactic (subject-verb agreement) and semantic information. However, there is a limit to which an RNN can effectively encode information given a long sequence. This is due the way in which they are trained. The *backpropagation* training algorithm suffers from the problem of vanishing gradients, where the gradients of the parameters become very small (tends to zero) when the sequences become very long. Long-short-term memory (LSTM) are a special type of RNNs which ameliorate the issue of vanishing gradients through a gating mechanism. The forget gates determine which words we can forget in a sequence and only focus on the important words. Since sequence in both directions are important, forward and backward, we use a bidirectional LSTM on the words of the context. We initialize

¹ Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.) *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc

Figure 3.1: Mean Teacher framework for the named entity classification task (left). E_{wi} are words in the entity mention, W_i are words in the context with entity mention replaced by $\langle E \rangle$ token. $cost = (classification_cost) + \lambda(consistency_cost)$. Unlabeled examples have only consistency cost. Backprop only through student model, teacher model parameters are averaged.

Figure 3.2: The architecture of the student or teacher model (right). Noise can be added to parts in shown boldface. They are (1) input noise (2) statistical noise (3) network dropout. $predictions = \text{softmax}(\text{output_layer})$

the words in the example with pre-trained word embeddings and run a bi-directional LSTM on both the entity mention and its context. We concatenate the final LSTM state of both the mention and the context representations and run a multi-layer perceptron with one hidden layer to produce the output layer.

Noise Strategies

We add a similar but distinct noise to both the teacher and the student models as seen in Figure 3.2 (parts in boldface).

1. *Input noise* – In the form of linguistically motivated noise such as *word dropout*, or *replacing words* with their synonyms (more details below).
2. *Statistical noise* – In the form of standard Gaussian perturbations to pre-trained word embeddings.

3. *Network noise* – Dropout in the intermediate layers of the student and teacher networks.

The idea of adding noise is to regularize the model parameters and help learn robust models in the scenario of very limited labeled training data using the teacher and student models via the consistency cost. Consequently, the MT framework can also be perceived as a consistency regularization technique.

The input noise is applied to the context of an entity mention. The noise was added to a fixed number of words in a context. We explored different types of *input noise*:

(1) *Word-dropout* - dropping words randomly in the input context. In the example below, named entity in the sentence is "The World Trade Organization and China Service Trade Seminar" is replaced with a special token called @ENTITY. In each of the types of noises we can add or replace any word other than the entity itself.

For example, Entity - " The World Trade Organization and China Service Trade Seminar " and Context - @ENTITY will be held in Shanghai on April 19th.

Word-dropout - @ENTITY will be held in Shanghai on 19th. (Here April is the word that has been randomly dropped)

(2) *Synonym-replace* - replace a randomly chosen word in the context by its synonym from WordNet. Following the same example above, any word in the context is randomly replaced by its synonym.

For example, Synonym-replace - @ENTITY will be keep in Shanghai on April 19th. (Here *held* has been picked randomly and is replaced by keep², since keep is among the top five synonyms for *held* in the WordNet dictionary)

² held is lemmatized to hold

(3) *Word-dropout-idf* - drop the most informative word in the context, as determined by the inverse document frequency (IDF) of context words computed offline. Following the same example above, word with high IDF score is dropped.

Word-dropout-idf - @ENTITY will be in Shanghai on April 19th. (Here *held* is the most informative word given by its IDF score has been dropped)

(4) *Synonym-replace-idf* - replace the words in the context according to their IDF (as described above). Following the same example above, word with high IDF score is replaced with its synonym from WordNet.

Synonym-replace-idf - @ENTITY will be keep in Shanghai on April 19th. (Here *held* is the most informative word given by its IDF score has been replaced by keep)

For the *statistical noise*, we perturbed the pre-trained word embeddings with standard Gaussian noise with a fixed standard deviation. We varied the amount of standard deviation and the number of words to which this type of noise is added. As we demonstrate in the experiments, this requires careful tuning. Further, adding Gaussian noise is a computationally intensive process

as we need to perform this operation in every minibatch during the training process.

We implemented network noise with dropout with fixed probability in both the context representation and the hidden layer in the multi-layer perceptron. This is a standard regularization strategy used in deep learning systems.

Finally, we combined network noise with input noise. Empirically, we show that this combination yields the best possible performance for the task addressed. We also showed that linguistic noise provides complementary information to network noise.

4 Experiments

To recapitulate we are designing our experiments to study the extent to which noise helps in semi-supervised deep learning. The goal here is to model noise and hence we have chosen a standard yet important task in natural language processing which is named entity classification. Our insights would be extendable to other tasks as well.

Task and Datasets

The task investigated in this work is named entity classification (NEC), defined as identifying the correct type of an entity mention in a given context, e.g., classifying “*Bill Clinton*” in the sentence “*Former President Bill Clinton expects to attend the inauguration tomorrow.*” as a person name. We define the context as the complete sentence in which the entity mention appears. We use standard benchmark datasets, namely, CoNLL-2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003)¹ and Ontonotes-2013 (Pradhan et al., 2013)². Our setting is semi-supervised NEC, so we randomly select a very small percentage of the training dataset (40 datapoints *i.e.* 0.18% of CoNLL and 440 datapoints *i.e.* 0.56% of Ontonotes as labeled data, and artificially remove the labels of the remaining datapoints to simulate the semi-supervised setting. Our task is to predict the correct labels of the unlabeled datapoints. CoNLL had 4 label categories while Ontonotes has 11. The histogram of the labels in CoNLL is shown in Figure 4.1 and the histogram of OntoNotes labels is shown in Figure 4.2. We measure the accuracy as the percentage of the datapoints which have been predicted with the correct labels.

Experimental Settings

We use the entity boundaries for all datapoints during training but only use labels for a small portion of the data as indicated above. We demonstrate an input to our model in the bottom-right of Figure 2.1. To reduce computational overhead, we filtered out entity mentions which were greater than length 5 from the Ontonotes dataset (4 respectively for CoNLL), and contexts which were greater than length 59 or smaller than length 5 (40 and 3 respectively for

¹ Tjong Kim Sang, E. F. and F. De Meulder (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Daelemans, W. and M. Osborne (eds.) *Proceedings of CoNLL-2003*, pp. 142–147. Edmonton, Canada

² Pradhan, S., A. Moschitti, N. Xue, and H. T. Ng (2013). Linguistic Analysis using OntoNotes. In *Proceedings Computational Natural Language Learning*, pp. 143–152. ACL, Sofia, Bulgaria

Figure 4.1: CoNLL dataset histogram of the labels

Figure 4.2: OntoNotes dataset histogram of the labels

CoNLL). Following Nagesh and Surdeanu (2018a)³, we initialized the pre-trained word-embeddings from Levy and Goldberg (2014)⁴ (300d). We ran a 100d bi-directional LSTM on both the entity and context representations, concatenated their outputs and fed them to a 300d multi-layer perceptron with ReLU activations. For network dropout we used $p = 0.2$. We tried a few variations of this model such as augmenting the LSTM with position embeddings, attention and replacing the LSTM with an average model, but did not observe a considerable improvement in performance.

Results

We present our main results in Table 4.1. An important note is that the results are the accuracy of classification over 21,373 and 78,492 datapoints in

³ Nagesh, A. and M. Surdeanu (2018a). An Exploration of Three Lightly-supervised Representation Learning Approaches for Named Entity Classification. In *COLING*

⁴ Levy, O. and Y. Goldberg (2014). Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland

		CoNLL	Ontonotes
<i>No noise</i>		65.76 (± 2.06)	64.20 (± 2.27)
<i>Word-dropout</i>	1 W	67.70 (± 2.97)	67.46 (± 3.53)
	2 W	68.15 (± 3.15)	68.19 (± 3.35)
	3 W	68.54 (± 3.38)	68.42 (± 3.94)
<i>Synonym-replace</i>	1 W	67.56 (± 3.04)	67.70 (± 3.20)
	2 W	67.95 (± 3.17)	68.40 (± 3.62)
	3 W	68.35 (± 3.07)	68.46 (± 4.06)
<i>Word-dropout-idf</i>	1 W	67.59 (± 3.03)	67.38 (± 3.29)
	2 W	68.11 (± 3.17)	68.14 (± 3.63)
	3 W	68.49 (± 3.27)	68.30 (± 3.77)
<i>Synonym-replace-idf</i>	1 W	67.51 (± 3.02)	67.24 (± 3.55)
	2 W	67.79 (± 3.15)	68.23 (± 3.42)
	3 W	68.26 (± 3.05)	67.95 (± 3.96)
<i>Gaussian (stdev=4)</i>	all W	62.98 (± 2.66)	64.89 (± 5.12)
<i>Network Dropout</i>		68.40 (± 3.11)	71.77 (± 2.18)

Table 4.1: Overall accuracies comparing all noise strategies on CoNLL and Ontonotes datasets. *No noise* is the baseline. $X\ W \Rightarrow X$ words perturbed by noise. Accuracy is % of correctly classified datapoints. ($\pm y$) $\Rightarrow$ variance of 5 runs.

Figure 4.3: Performance upon combining noise strategies, CoNLL. Best performance: *network dropout + 5W word-dropout* - 70.57% (CoNLL)

CoNLL and Ontonotes respectively, using only a tiny sliver of the labels in these datasets as supervision. We report the average (along with their variance) of 5 randomized runs in each noise setting. Our baseline is the *no noise* setting, where the input to the student and teacher models are not augmented by noise.

From Table 4.1, we observe that adding noise is necessary for good performance, as we see that the various noise strategies consistently improve performance over the baseline on both the datasets. Network noise is a crucial factor for good performance. Input noise which are linguistically motivated, such as *word-dropout* and *synonym-replace* perform as well as the statistical noise. More specifically, *word-dropout* of 3 words and *synonym-replace* of 3 words, are the highest performing non-network noise strategies on CoNLL and Ontonotes respectively. Adding *Gaussian* noise to all words results in performance poorer than or close to baseline. Furthermore, *Gaussian* noise requires fine-tuning over the value of stdev and the number of words on

Figure 4.4: Performance upon combining noise strategies Ontonotes. Best performance: *network dropout + 3W synonym-replace* - 72.78% (Ontonotes)

Ontonotes - Accuracy		
	CoNLL	Ontonotes
1 W	69.70 (± 2.93)	68.75 (± 3.02)
5 W	68.48 (± 2.65)	68.22 (± 3.45)
10 W	66.55 (± 4.20)	67.32 (± 3.42)
stdev=0.05	68.51 (± 3.13)	68.42 (± 4.15)
stdev=1	66.94 (± 2.59)	66.79 (± 3.67)
stdev=2	65.43 (± 2.68)	65.90 (± 4.35)
stdev=4	62.98 (± 2.66)	64.94 (± 5.91)
stdev=8	62.49 (± 2.76)	64.02 (± 4.92)
stdev=16	62.87 (± 3.08)	64.85 (± 6.25)

Table 4.2: Tuning Gaussian noise - #words & stdev

which these should be applied which makes this computationally expensive approach (Table 4.2). The table also shows that for stdev values 8 and 16, the accuracies go below no-noise which is the baseline in our case suggesting that tuning Gaussian noise is critical for attaining good performance. The performance on **-idf* runs suggest that random word selection is as good or better. This is ideal, since it is simpler and independent of the data distribution. Finally, *network noise* in combination with linguistic input noise provides the best possible performance, as seen in Figure 4.3 and Figure 4.4. Also Table 4.3 shows accuracy and stdev of hybrid noise on both the datasets. We see in the table that hybrid noise performs better than individual noise strategies. Linguistic noise gives complementary information along with network dropout to give better performance for the model. Also we see in the table that all noise strategies perform better than no-noise condition which is expected because a regularized model is more robust and generalizes better on unseen data.

Discussion

The linguistic noise strategies are added before the input to the training algorithm. So the whole training process and the model can be treated as a black box. This not only modularizes the system it also helps in faster training as the noise augmentation sits outside the training. On the other

	CoNLL	Ontonotes
No-noise	65.76 (± 2.06)	64.20 (± 2.27)
network dropout	68.4 (± 3.11)	71.77 (± 2.18)
network dropout + 3W synonym-replace	69.72 (± 3.33)	72.78 (± 2.94)
network dropout + 5W synonym-replace	70.21 (± 3.98)	72.16 (± 3.57)
network dropout + 3W word-dropout	70.57 (± 3.79)	72.22 (± 3.32)

Table 4.3: Performance upon combining noise strategies on CoNLL and Ontonotes with accuracy and stdev

hand, statistical noise where we need to perturb the word embeddings, are much more involved to implement because during each forward training step we need to add a random delta noise and this happens in each mini batch. Thus increasing the training time to twice or three times depending on how many words we need to perturb. Sacrificing modularity also costs increased train times for the model. While linguistic noise being simpler gives similar performance and they is more interpretable in some sense. Say in case of replacing a word with its synonym is equivalent to perturbing the embedding in a similar space as that of the word in the high dimensional space. But when we randomly perturb, as adding Gaussian noise, we are not sure of its orientation and this makes the input noise less interpretable.

5 *Conclusion and Future Work*

We summarize the salient points of our work as follows. Semi-supervised learning reduces the dependence on annotated training data. Bootstrapping-based semi-supervised learning suffers from the problem of semantic drift. Mean teacher architecture is one shot learning framework for semi-supervised learning and performs better than bootstrapping based approaches since it reduces semantic drift. Noise is an important part of semi-supervised learning in the mean teacher framework. However, noise has received limited attention from the natural language processing community. We address this gap by exploring several noise strategies for the semi-supervised named entity classification task. Further, we show that linguistic noise such as word-dropout and synonym-replace perform as well as statistical noise, while being simpler and easier to tune. A combination of linguistic and network dropout provides the best performance.

Our work demonstrates that an inter-disciplinary approach combining linguistics and statistical noise is a promising direction. The exploration of noise for the semi-supervised learning for NLP explored in this work will be helpful to generalize to other areas of NLP.

As part of future work, we wish to explore noise augmentation in other language processing tasks such as fine-grained entity typing and relation extraction. Since we explored named entity classification and in one of the datasets we had 11 classes for named entity classification, it would be a natural extension for us to try on fine-grained entities where there are more number of types for the same entity depending on the context. For example the entity, *Barack Obama*, can have the label *president* or *book author* based on its context. Another related NLP application is relation extraction where we are trying to predict the relation between two entities. This task is called relation extraction. For example between the entities say *Barack Obama* and *Michelle Obama* the relationship is *spouse*. It is a more complicated task when compared to named entity classification.

References

- Abney, S. (2007). *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 1st edition. ISBN 1584885599, 9781584885597.
- Belinkov, Y. and Y. Bisk (2017). Synthetic and Natural Noise Both Break Neural Machine Translation. *CoRR*, **abs/1711.02173**.
- Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell (2010a). Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.
- Carlson, A., J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell (2010b). Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 101–110. ACM.
- Clark, K., T. Luong, C. D. Manning, and Q. V. Le (2018). Semi-Supervised Sequence Modeling with Cross-View Training.
- Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02*, pp. 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA. DOI: 10.3115/1118693.1118694.
- Collins, M. and Y. Singer (1999). Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. The MIT Press. ISBN 0262035618, 9780262035613.
- Goodfellow, I. J., J. Shlens, and C. Szegedy (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Gupta, S. and C. D. Manning (2014). Improved Pattern Learning for Bootstrapped Entity Extraction. In *CoNLL*, pp. 98–108.
- Gupta, S. and C. D. Manning (2015). Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.
- Komachi, M., T. Kudo, M. Shimbo, and Y. Matsumoto (2008). Graph-based Analysis of Semantic Drift in Espresso-like Bootstrapping Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '08*, pp. 1011–1020. Association for Computational Linguistics, Stroudsburg, PA, USA.

- Laine, S. and T. Aila (2016). Temporal Ensembling for Semi-Supervised Learning. *CoRR*, **abs/1610.02242**.
- Levy, O. and Y. Goldberg (2014). Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland.
- McIntosh, T. (2010). Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 356–365. Association for Computational Linguistics.
- Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.) *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc.
- Mitchell, T. M. (1997). *Machine learning*. McGraw Hill series in computer science. McGraw-Hill. ISBN 978-0-07-042807-2.
- Miyato, T., S. ichi Maeda, M. Koyama, and S. Ishii (2018a). Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning. *IEEE transactions on pattern analysis and machine intelligence*.
- Miyato, T., S. Maeda, S. Ishii, and M. Koyama (2018b). Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1. ISSN 0162-8828. DOI: 10.1109/TPAMI.2018.2858821.
- Moindrot, O. and G. Genthial (2018). Splitting into train, dev and test sets: Best practices to split your dataset into train, dev and test sets. <https://cs230-stanford.github.io/train-dev-test-split.html>. Accessed: 2018-04-03.
- Nagesh, A. and M. Surdeanu (2018a). An Exploration of Three Lightly-supervised Representation Learning Approaches for Named Entity Classification. In *COLING*.
- Nagesh, A. and M. Surdeanu (2018b). Keep your bearings: Lightly-supervised Information Extraction with Ladder Networks that avoids Semantic Drift. In *NAACL HLT 2018*.
- Pradhan, S., A. Moschitti, N. Xue, and H. T. Ng (2013). Linguistic Analysis using OntoNotes. In *Proceedings Computational Natural Language Learning*, pp. 143–152. ACL, Sofia, Bulgaria.
- Raj, B. (2018). Data Augmentation - How to use Deep Learning when you have Limited Data - Part 2. <https://bit.ly/2IvKw11>. Accessed: 2018-12-10.
- Rasmus, A., H. Valpola, M. Honkala, M. Berglund, and T. Raiko (2015). Semi-Supervised Learning with Ladder Network. *CoRR*, **abs/1507.02672**.
- Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pp. 1044–1049.
- Sammut, C. and G. I. Webb (2011). *Encyclopedia of Machine Learning*. Springer Publishing Company, Incorporated, 1st edition. ISBN 0387307680, 9780387307688.
- Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**, pp. 1929–1958.

- Tarvainen, A. and H. Valpola (2017). Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, **abs/1703.01780**.
- Tjong Kim Sang, E. F. and F. De Meulder (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Daelemans, W. and M. Osborne (eds.) *Proceedings of CoNLL-2003*, pp. 142–147. Edmonton, Canada.
- Yangarber, R. (2003). Counter-training in discovery of semantic patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- Zhu, X. (2005). Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.
- Zhu, X. and Z. Ghahramani (2002). Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.