Zoho at SemEval-2019 Task 9: Semi-supervised Domain Adaptation using Tri-training for Suggestion Mining

Sai Prasanna
Zoho
saiprasanna.r@zohocorp.com
sai.r.prasanna@gmail.com

Sri Ananda Seelan
Zoho
anandaseelan.ln@zohocorp.com

Abstract

This paper describes our submission for the SemEval-2019 Suggestion Mining task. A simple Convolutional Neural Network (CNN) classifier with contextual word representations from a pre-trained language model is used for sentence classification. The model is trained using tri-training, a semi-supervised bootstrapping mechanism for labelling unseen data. Tri-training proved to be an effective technique to accommodate domain shift for cross-domain suggestion mining (Subtask B) where there is no hand labelled training data. For in-domain evaluation (Subtask A), we use the same technique to augment the training set. Our system ranks thirteenth in Subtask A with an F_1-score of 68.07 and third in Subtask B with an F_1-score of 81.94.

1 Introduction

Task 9 of SemEval-2019 (Negi et al., 2019) focuses on mining sentences that contain suggestions in online discussions and reviews. Suggestion Mining is modelled as a sentence classification task with two Subtasks:

- Subtask A evaluates the classifier performance on a technical domain specific setting.
- Subtask B evaluates the domain adaptability of a model by doing cross-domain suggestion classification on hotel reviews.

We approached this task as an opportunity to test the effectiveness of transfer learning and semi-supervised learning techniques. In Subtask A, the high class imbalance and relatively smaller size of the training data made it an ideal setup for evaluating the efficacy of recent transfer learning techniques. Using pre-trained language models for contextual word representations has been shown to improve many Natural Language Processing (NLP) tasks (Peters et al., 2018; Ruder and Howard, 2018; Radford, 2018; Devlin et al., 2018). This transfer learning technique is also an effective method when less labelled data is available as shown in (Ruder and Howard, 2018). In this work, we use the BERT model (Devlin et al., 2018) for obtaining contextual representations. This results in enhanced scores even for simple baseline classifiers.

Subtask B requires the system to not use manually labelled data and hence it lends itself to a classic semi-supervised learning scenario. Many methods have been proposed for domain adaptation for NLP (Blitzer et al., 2007; Chen et al., 2011; Chen and Cardie, 2018; Zhou and Li, 2005; Blum and Mitchell, 1998). We use a label bootstrapping technique called tri-training (Zhou and Li, 2005) with which unlabelled samples are labelled iteratively with increasing confidence at each training iteration(explained in Section 2.4). Ruder and Plank (2018) shows the effectiveness of tri-training for baseline deep neural models in text classification under domain shift. They also propose a multi-task approach for tri-training, however we only adapt the classic tri-training procedure presented for suggestion mining task.

Detailed explanation of the submitted system and experiments are elicited in the following sections. Section 2 describes the components of the system. Following this, Section 3 details the experiments, results and ablation studies that were performed.

2 System Description

The models and the training procedures are built using AllenNLP library (Gardner et al., 2018). All the code to replicate our experiments are public and can be accessed from https://github.com/sai-prasanna/suggestion-mining-semeval19.

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1282–1286
Minneapolis, Minnesota, USA, June 6–7, 2019.

2.1 Data cleaning and pre-processing

Basic data pre-processing is done to normalize whitespace, remove noisy symbols and accents. Very short sentences with less than four words are disregarded from training.

2.2 Word Representations

We use GloVe word representations (Pennington et al., 2014) and compare the performance improvement that we obtain with pre-trained BERT representations (Devlin et al., 2018).

2.3 Suggestion Classification

Our baseline classifier is Deep Averaging Network (DAN) (Iyyer et al., 2015). DAN is a neural bag-of-words model that is considered as a strong baseline for text classification. In DAN, a sentence representation is obtained by averaging the word level representations and is fed to a series of rectified linear unit (ReLU) layers with a final softmax layer.

A simple Convolutional Neural Network (CNN) text classifier (Kim, 2014) is used for the final submission.

2.4 Training

We use the classic tri-training procedure for label bootstrapping as mentioned in (Ruder and Plank, 2018). Consider a labelled dataset L from the source domain S and an unlabelled dataset U from the target domain T. The objective of tri-training is to label U iteratively and augment it with L. Three CNN +BERT classifiers M_1, M_2, M_3 are trained separately using subsets of L namely l_1, l_2, l_3 respectively. These subsets are obtained from L using bootstrap sampling with replacement.

The above mentioned models are used to predict labels for the unlabelled set U. Predictions which are agreed by two models is considered as a new training example for the third model in the next iteration. For example, an unlabelled sentence $U_1 \in U$ is added as a labelled example to l_1, if and only if the label for U_1 is agreed upon by both M_2 and M_3. Same way, l_2 is updated with newly labelled data if those labels have been agreed by M_1 and M_3 and so on. This constitutes a single iteration of tri-training. The procedure that is used for the training of our models is mentioned in Algorithm 1.

In this way, the original training data gets added with three newly labelled subsets which are again used for the next training iteration. At the end of each iteration, validation F_1-score is calculated by using the predictions that are obtained through a majority vote. The procedure is continued until there is no improvement in the validation score.

Algorithm 1 Tri-training

$L \leftarrow Labelled\ Data\ ,\ |L| = m$
$U \leftarrow Unlabelled\ Data\ ,\ |U| = n$
for $i \leftarrow 1, 2, 3$ **do**
    $l_i \leftarrow BootstrapSamples(L)$
end for
repeat
    for $i \leftarrow 1, 2, 3$ **do**
        $M_i \leftarrow Train(l_i)$
    end for
    for $i \leftarrow 1, 2, 3,$ **do**
        $l_i \leftarrow L$
        for $j \leftarrow 1, n$ **do**
            if $M_p(U_j) == M_q(U_j)$
                $where\ p, q \neq i$ **then**
                $l_i \leftarrow l_i + \{(U_j, M_p(U_j)\}$
            end if
        end for
    end for
until no improvement in validation metrics

3 Experiments and Results

This section details the various experiments that were performed using the above components for our submissions.

3.1 Data

The test set provided during the trial phase of the evaluation is used as the validation data for all our experiments. For those experiments that do not involve tri-training, we only use the provided labelled data from the technical domain for training.

In Subtask B, for those experiments that involve tri-training, L is the same as mentioned above. U here is obtained in two ways:

- Unlabelled data from final test set of Subtask B.
- Unlabelled data from Yelp hotel reviews (Blomo et al., 2013).

The results reported are mean and confidence intervals of Precision, Recall and F_1-score over five runs of the same experiments with different random seeds.

Experiment	Validation			Test		
	Precision	Recall	F_1	Precision	Recall	F_1
Subtask A - Technical Domain						
Organizer Baseline	58.72	93.24	72.06	15.69	91.95	26.80
DAN +glove	68.51±2.43	87.30±5.00	76.69±1.06	25.40±3.56	84.60±9.87	38.84±3.10
DAN +bert	76.06±1.31	90.27±1.71	82.55±0.50	45.80±4.49	90.80±1.75	60.82±3.99
DAN +bert w/o upsampling	79.04±2.67	83.38±2.73	81.11±0.68	55.06±6.36	83.68±2.75	66.28±4.28
CNN +bert	80.34±4.21	89.93±4.23	84.76±0.52	50.34±6.70	91.72±2.55	64.81±4.86
CNN +bert w/o upsampling	83.22±3.01	84.73±3.86	83.90±0.70	58.98±5.41	88.05±1.63	**70.58**±4.24
CNN +bert +tritrain$_{Test}$*	83.06±1.96	89.19±1.88	**86.00**±0.35	52.89±2.69	90.80±2.02	66.81±1.90
Subtask B - Hotel Reviews Domain						
Experiment	Validation			Test		
	Precision	Recall	F_1	Precision	Recall	F_1
Organizer Baseline	72.84	81.68	77.01	68.86	78.16	73.21
DAN +glove	82.00±4.25	52.97±9.25	64.01±5.75	73.32±3.50	46.09±7.21	56.35±4.71
DAN +bert	89.75±2.79	65.74±8.71	75.65±5.10	78.90±4.03	64.20±8.77	70.49±4.09
DAN +bert w/o upsampling	94.26±1.87	31.73±5.73	47.31±6.27	87.98±3.41	31.09±7.17	45.62±7.47
CNN +bert	93.77±1.34	51.88±6.88	66.65±5.68	90.17±2.45	50.34±8.71	64.31±6.72
CNN +bert w/o upsampling	93.94±1.36	45.99±7.59	61.53±6.73	89.75±4.41	44.08±9.38	58.66±7.79
CNN +bert +tritrain$_{Test}$*	91.91±2.06	88.32±2.05	**90.05**±0.76	81.26±1.63	83.16±1.40	**82.19**±1.03
CNN +bert +tritrain$_{Yelp}$	88.09±0.62	87.13±0.38	87.61±0.42	78.01±5.42	86.67±3.96	81.98±2.05

Table 1: Performance metrics of different models on validation and test sets of both subtasks. Confidence intervals for the metrics are reported for five runs using different random seeds on t-distribution with 95% confidence. Upsampling is used in the training dataset unless otherwise specified. Single model from experiments with * was used for the final submission.

3.2 Input

For input representations, we use 300d GloVe vectors with dropout (Srivastava et al., 2014) of 0.2 for regularization. We also experiment with the pre-trained BERT base uncased model. The BERT model is not fine-tuned during our training. A dropout of 0.5 is applied for the 768d representations obtained from BERT.

3.3 Baseline Deep Averaging Net

Our neural baseline is Deep Averaging Net (DAN) (Section 2.3). When used with GloVe, the hidden sizes of DAN are 300, 150, 75, and 2 respectively. When BERT representations are used, the hidden sizes of the network are 768, 324, 162, and 2 respectively. We report an F_1-score of 60.82 when DAN is used with BERT in Subtask A and 70.49 in Subtask B. Both these scores are a significant improvements from those obtained with GloVe representations (Table 1).

We retain the same configuration of BERT embedding layer for other experiments also. Training is performed with Adam (Kingma and Ba, 2015) optimizer with a learning rate of $1e{-}3$ for all the models.

3.4 CNN Classifier

The CNN classifier is composed of four 1-D convolution layers with filter widths ranging from two to five. Each convolutional layer has 192 filters. The output from each layer is max-pooled over sequence (time) dimension. This results in four 192d vectors, which are concatenated to get a 768d output.

The max-pooled outputs are passed through four fully connected feed forward layers with hidden dimensions of 768, 324, 162, and 2 respectively. The intermediate layers use ReLu activation and the final layer is a softmax layer. We use dropout of 0.2 on all layers of the feed forward network except for the final layer.

Without tri-training, this model obtains an absolute improvement of $\approx 4\%$ F_1-score over DAN in Subtask A. However in Subtask B, it performs poorer than the baseline DAN model with an F_1-score of 64.31. This decrease in performance could be because of overfitting on the source domain due to the larger number of parameters in CNN compared to DAN.

3.5 Tri-training

The aim of doing tri-training is for domain adaptation by labelling unseen data from a newer domain. For Subtask B, the CNN + BERT model achieves an F_1-score of 82.19 when trained with the tri-training procedure mentioned in Algorithm 1. Tri-training is used to label the 824 unlabelled sentences from the test set of Subtask B and augmented with the original training data. This score is a huge improvement from the classifier model trained only on the given data which gets an F_1-score of 64.31.

We also do the same experiment using 5000 unlabelled sentences from Yelp hotel reviews dataset (Blomo et al., 2013). The model obtains a similar score of 81.98 which proves the importance of tri-training in domain adaptation.

For Subtask A, we get an improvement in the F_1-score using tri-training, however the increase is not as profound as we observe for Subtask B. We compare the statistical significance of the different models and experiments in Section 3.7.

3.6 Upsampling

We also wanted to find how the class balance in the dataset has affected our model performance. The class distribution of the datasets including the test set distribution that was obtained after the final evaluation phase are mentioned in Table 2.

Dataset	Suggestions (%)
Training	23
Subtask A validation	50
Subtask B validation	50
Subtask A Test	**10**
Subtask B Test	42

Table 2: Label distribution

The original training data has a class imbalance with only 23% of the sentences labelled as suggestions. We tried to balance the labels by naive upsampling, ie., adding duplicates of sentences that are labelled as suggestions. This allowed us to have a balanced training dataset for our experiments. This resulted in consistent gains over the original dataset during the trial evaluation phase.

However during the final submission, in Subtask A we found that the model's performance in the test set did not correlate well with that of the validation set as shown in Table 1. This could be because the percentage of positive labels in the test set is only 10% while the validation set has 50%.

Experiments without upsampling gives better performance in test set even though there is a decrease in the validation score. For Subtask B however, upsampling has actually increased the model performance. On hindsight, this could be because of similar distribution of class labels in both validation and test sets.

The submitted models received an F_1-score of 68.07 in Subtask A and 81.03 in Subtask B.

3.7 Statistical Significance Test

Reichart et al. (2018) suggests methods to measure whether two models have statistically significant differences in their predictions on a single dataset. We incorporate a non-parametric testing method for significance called the McNemar's test recommended by them for binary classification. Pairwise comparison of few of our models are reported in Table 3. The table contains the p-values for the null hypothesis. The null hypothesis is that two models do not have significant differences in their label predictions. In simpler words, a small p-value for an experiment pair denotes a significant difference in the prediction disagreement between two models. For example, from Table 3, DAN + GloVe and DAN + BERT models have a p-value less than 0.05 in both sub-tasks. This indicates that there is significant disagreement between the predictions of two models. Since DAN + BERT gets a better F_1-score and $p < 0.05$, we can confidently assert that improvement is not obtained by chance.

We use majority voting from five random seeds to get the final predictions on the test set for doing the paired significance testing.

4 Conclusion

We discussed our experiments for doing suggestion mining using tri-training. Tri-training combined with BERT representations proved to be an effective technique for doing semi-supervised learning especially in a cross-domain setting. Future work could explore more optimal ways of doing tri-training, evaluate the effect of contextual representations in tri-training convergence, and try more sophisticated architectures for classification that may include different attention mechanisms.

Subtask	Model A	Model B	p-value
A	DAN +glove	DAN +bert	≈ 0
A	DAN +bert	CNN +bert	0.046
A	CNN +bert	CNN +bert +tritrain$_{Test}$	**0.108**
B	DAN +glove	DAN +bert	$1.419e-05$
B	DAN +bert	CNN +bert	**0.4208**
B	CNN +bert	CNN +bert +tritrain$_{Test}$	$3.251e-08$
B	CNN +bert +tritrain$_{Test}$	CNN +bert +tritrain$_{Yelp}$	**0.5862**

Table 3: Pairwise comparison of various models using the McNemar's Test. $p \leq 0.05$ indicates a significant disagreement between the model predictions.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Jim Blomo, Martin Ester, and Marty Field. 2013. Recsys challenge 2013. In *RecSys*.

Avrim Blum and Tom M. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

Minmin Chen, Kilian Q. Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS*.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Roi Reichart, Rotem Dror, Gili Baumer, and Segev Shlomov. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *ACL*.

Sebastian Ruder and Jeremy Howard. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17:1529–1541.