Evaluating Domain Adversarial Neural Networks on Multi-Genre Natural Language Inference

Mor Geva
Tel-Aviv University
morgeva@mail.tau.ac.il

Nadav Schor
Tel-Aviv University
nadavschor@mail.tau.ac.il

Meishar Meiri
Tel-Aviv University
meishar@gmail.com

Abstract

The recently published MULTINLI dataset introduces the domain adaptation challenge for the Natural Language Inference (NLI) task. In this work, we address this challenge by using a domain adversarial approach, which has been previously applied for multiple Natural Language Processing (NLP) tasks. We extend the setup used in prior work, to support training with multiple source domains, compare between the two baselines and analyze the adversarial component performance. Finally, we conduct a feature-based analysis of the data, considering the relation between key features of each domain and the performance of the adversarial approach.

1 Introduction

In the task of Natural Language Inference (NLI), a model is given a pair of sentences, premise and hypothesis, and is asked to predict their inference relation: entailment, contradiction or neutral. The task is simply defined but consist of many Natural Language Processing (NLP) aspects and so requires the model to comprehend the content and the meaning of the sentences. Concretely, the model needs to cope with different ways of expression such as first/third person, tense, slang and more.

Two major datasets were presented in the last few years, for evaluation of the NLI task. Stanford Natural Language Inference (SNLI) [1] is a known dataset with 570k human-written pairs of sentences, which has been used for evaluation of various models. The premise is a caption of a photo, taken from the FLICKR30K dataset [7]. The hypothesis was produced by an Amazon Mechanical Turk worker, that given only a caption, was tasked to write three sentences with regards to it, one for each label. Recently, Multi-Genre Natural Language Inference (MULTINLI) [10] was published as part of the RepEval 2017 shared task[1]. This dataset consists of ten different text genres, with 433k pairs of sentences in total. While the hypothesis sentences were produced as in SNLI, the premise sentences were collected from ten different textual sources. The ten genres vary in content, from formal government documents to transcript of face-to-face conversations. MULTINLI is divided into two genre groups, MATCHED and MISMATCHED. The MATCHED group contains five genres, each has a training set of $\sim$77k samples, a development set of 2k samples and a test set of 2k samples. The MISMATCHED group, holds only development and test sets. All the test sets were published without labels.

Domain adaptation is a field in machine learning, that aspires to bridge between a target domain, with

[1] https://repeval2017.github.io

unlabeled data, and a source domain with labeled data. Recent domain adaptation methods try to ease the shift between the source and target domains by embedding the target samples into the same feature space as the source domain. Domain adversarial training is a specific domain adaptation approach, which uses a new adversarial component called domain classifier, to minimize the distance between the source domain and the target domain distributions. The domain classifier operates against the feature extractor component of the model, while trying to discriminate between the domains. This approach was first introduced in the field of computer vision, and in the last few years, was presented in the field of NLP as well [3, 4, 8, 11]. Different adversarial methods were suggested as well, for example, weight sharing across domains for learning symmetric representations [3, 8, 11].

The release of MULTINLI enables to address the domain adaptation challenge for NLI. In this work, we apply and extend the domain adversarial approach for this task. We propose a novel modification, which allows the usage of multiple source domains, in order to obtain a better generalization of the model. We evaluate our method on both SNLI and MULTINLI.

2 Related Work

Many novel NLI methods were presented since the publication of SNLI, currently the state-of-the-art model achieves accuracy of 88.6% on the test set [2], and 88.8% for an ensemble [9]. Three baseline models were presented for the MULTINLI dataset, achieving accuracy of 67.5% on the MATCHED and 67.1% on the MISMATCHED. Yet, neither any of the models applied to SNLI, nor the MULTINLI baselines directly address the domain adaptation aspect of the NLI task. In our work, we use both datasets, together comprised of almost 1M samples, to evaluate the domain adversarial training approach.

A few variations of the domain adversarial training were suggested in the last years: Ganin and Lempitsky [3] proposed the use of three components: (1) Feature Extractor (FE), which receives the input and create a vector representation of the sample; (2) Label Predictor (LP), which receives the vector from the FE and predict the label; (3) Unsupervised Domain Classifier (DC), which receives the vector from the FE and classify the sample as one of the two domains. The first two parts form a standard feedforward network. In this approach, the network is trained end-to-end. The gradients of the DC multiply by a negative constant and added to the FE gradient at backpropagation. Ganin et al. [4] suggested an almost identical method, with a single difference: at backpropagation, the gradient of the DC is also multiplied by the same constant. This change decreases the learning rate of the DC. Tzeng et al. [8] introduced a new training method with three stages: (1) train a neural network composed of FE and LP, on the source domain. (2) remove the LP and connect a DC to the trained FE. In addition, randomly initial a second FE, that process the target samples, and connect it to the DC as well. Train the new FE using the DC. (3) at test time, connect the source LP to the target FE, and predict based on the LP predictions.

In this work, we followed Ganin and Lempitsky [3]. Overall, current approaches are using a single source domain and a single target domain [3, 4, 8, 11]. By doing so, they strict themselves to a single data source. Our approach enables the use of several source domains, thus making it possible to increase the size of the labeled data available for learning.

3 Model

We are given L labeled samples $\{(x_p, x_h)_l^d, y_l^d\}_{l \in L, d \in S}$ from a set of source domains S, and U unlabeled samples $\{(x_p, x_h)_l^d\}_{l \in U, d \in T}$ from a set of target domains T. We wish to predict the label class for every unlabeled sample.

3.1 Our Approach

We follow the domain adversarial training approach, as described in [3], with three components: a feature extractor, a label predictor and a domain classifier. The input of the network goes through the feature extractor, where the premise and the hypothesis pairs are encoded into a vector representation with bidirectional LSTM. The resulted vectors then go through both the label predictor and the domain classifier, in parallel. While the label predictor considers only the labeled samples $\{(x_p, x_h)_l^d, y_l^d\}_{l \in L, d \in S}$ to produce label predictions, the domain classifier considers both the labeled and the unlabeled samples. Learning is done using backpropagation with min-max optimization, where we minimize the label predictor loss and maximize the domain classifier loss. Figure 1 describes a high-level overview of the network. In test time, the input samples go through the feature extractor and the label predictor, for label prediction. Namely, the domain classifier is used only during training.

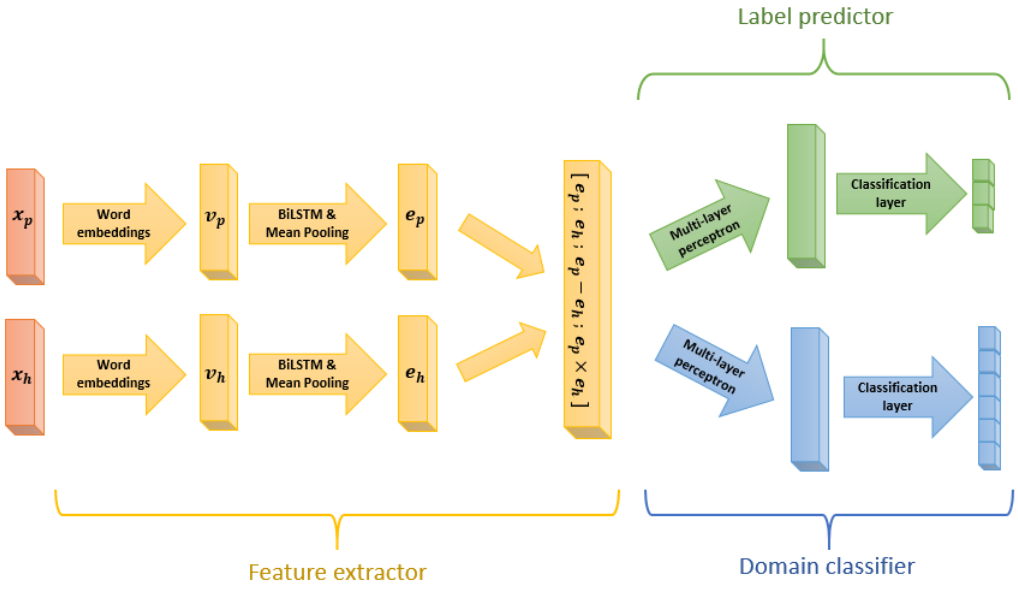

Figure 1: A high-level overview of the network architecture. Before entering the network, x_p and x_h are preprocessed. Details are given in section 3.2

3.2 Architecture Details

We first describe the label predictor and domain classifier components, which receive the output vector of the feature extractor, and then we describe the feature extractor itself.

Label predictor The label predictor receives the feature extractor output vector, and passes it through a multi-layer perceptron and a classification layer, that outputs a 3-dimensional vector. The resulted vector goes through a nonlinear activation function (ReLU), on which we apply dropout with rate ρ. To get a label prediction, softmax is applied and a cross-entropy loss L_l is computed. It is important to notice that only the labeled samples go through the label predictor.

Domain classifier The domain classifier receives the feature extractor output vector. It has the exact same structure as the label predictor, except for the classification layer, which has a 6-dimensional output vector (instead of 3-dimensional). Both the labeled and the unlabeled samples pass through the domain

classifier, which classifies them to one of the six possible domains (5 labeled domains and 1 unlabeled domain). The corresponding cross-entropy loss is denoted by L_d.

Feature extractor Here, we follow the RepEval BiLSTM baseline (see section 3.3) and [5]. The feature extractor component receives input samples, each is a parsed pair of premise and hypothesis sentences, denoted by x_p and x_h. Before entering the network, each of the sentences is in an unlabeled binary-branching format [10], which is padded (or clipped) to a constant length. Entering the network, x_p, x_h are first being transformed to a vectorized form, using GloVe embeddings[2] [6]. Here, we apply dropout with rate ρ on the words embeddings. Each of the two resulting vectors v_p, v_h, then goes through a bidirectional LSTM, after which a mean pooling is applied to get an element-wise mean over the output vectors. At this point, we have a separated encoding for the two input sentences e_p, e_h. The final output vector, which encodes the relation between the premise and the hypothesis, is defined as the vector $[\, e_p \,;\, e_h \,;\, e_p - e_h \,;\, e_p \times e_h \,]$. In words, the output vector is a concatenation of the mean pooling of the premise and the hypothesis encodings, the difference between them and the product between them. The loss L_f with respect to the feature extractor parameters is defined as $L_f = L_l - \lambda \cdot L_d$, where L_l and L_d are the label loss and domain loss, respectively. λ is a parameter that controls the weight of the domain classifier during training.

3.3 Implementation

Our implementation is based on the BiLSTM baseline code[3], which was published during the RepEval 2017 challenge. The code uses the Tensorflow package and it is publicly available on Bitbucket at `https://bitbucket.org/mega2/dann-mnli`. We also used the gradient flip operation, as implemented by @pumpikano[4]. Following prior work [3, 11], we gradually increase the weight of the discriminator (the domain classifier), where we denote by γ the growth rate of λ. We use the Adam algorithm for a stochastic gradient descent optimization, together with an exponential learning rate decay. In every batch, we balance the number of labeled and unlabeled samples, according to the number of represented domains. For example, in a batch that consists of 5 (4) labeled domains and 1 (2) unlabeled domain, the number of labeled samples will be 5 (2) times greater than the number of unlabeled samples. Before every epoch, the labeled and unlabeled samples are shuffled. Finally, we choose to fix part of the models parameters: we use initial learning rate of $1 \cdot e^{-4}$ and sequence length 50. The batch size is fixed to 80 labeled samples, where the number of unlabeled samples is determined per model as described above.

4 Experiments

4.1 Datasets and Evaluation Metrics

Since the MULTINLI test set is not publicly available with labels, we use the MULTINLI MATCHED and MISMATCHED development sets as our test sets, and split the original MATCHED training set and MISMATCHED development set to generate new development sets. The new sets were generated by randomly splitting the original sets, as detailed in Table 1. As the SNLI dataset is available with labels, we use the original sets as is.

[2] 840B tokens, 300d vectors

[3] `https://github.com/woollysocks/multiNLI`

[4] `https://github.com/pumpikano/tf-dann`

Dataset	Genre	Train set	Dev. set	Test set
	Origin:	Train SNLI	Dev SNLI	Test SNLI
SNLI		550152	10000	10000
	Origin:	Train MULTINLI	Train MULTINLI	Dev MATCHED
MULTINLI MATCHED	FICTION	75348	2000	2000
	TRAVEL	75350	2000	2000
	GOVERNMENT	75350	2000	2000
	TELEPHONE	81348	2000	2000
	SLATE	75306	2000	2000
	Origin:		Dev MISMATCHED	Dev MISMATCHED
MULTINLI MISMATCHED	9/11	-	1000	1000
	F2F	-	1000	1000
	LETTERS	-	1000	1000
	OUP	-	1000	1000
	VERBATIM	-	1000	1000

Table 1: Number of samples in SNLI and MULTINLI, with partition to train, development and test sets, as used in our experiments. Above every dataset, there is a row that describes from which original set (train/dev/test) the relevant samples were taken from.

We use classification accuracy as our main evaluation metric. We chose the hyperparameters, based on the model performance on the development sets. For the chosen hyperparameters we evaluate the model on the test sets. For stopping criteria, we use pre-defined number of epochs, while marking an early stopping (ES) point during training.

4.2 Experiment Setup

We examined our approach on two baselines. The first baseline is DANN2, where there is a single source domain and a single target domain. We trained 30 different models, for every combination of source-target domains from MULTINLI MATCHED and SNLI, using dropout rate 0.2 and $\gamma = 2$ across 32 epochs. As a control, for every source domain an additional model with zero weight to the domain classifier was trained, by setting $\gamma = 0$. Notice that these models are independent of the choice of target domain. Furthermore, we examined the effect growth rate has on generalization. We selected 7 domain combinations on which the adversarial approach improved the accuracy, and trained an additional model with $\gamma = 5$ for each one of them.

The second baseline is DANN5_1UL, where there are 5 source domains and a single target domain. While DANN2 is similar to previous reported experiments, DANN5_1UL is a new baseline, where the model is trained jointly on multiple source domains. Due to the limited number of samples in the MULTINLI MISMATCHED genres (see Table 1), for this baseline the SNLI was used as the unlabeled target domain. Notice that we balance the number of unlabeled samples in every batch, as described in section 3.3, so there is no bias while training the domain classifier. We examined the influence of both the growth rate and the dropout rate on generalization. First, three models were trained with $\gamma = 0, 2, 5$, while fixating the dropout rate to 0.2 and the number of epochs to 16. Then, another three models were trained with dropout rates 0, 0.2, 0.5 and $\gamma = 2$ fixed, across 16 epochs. For all the models described thus far, the three network components were trained simultaneously. As an additional experiment, we tried a second training method of two-step training (2ST). In this setup, the model is trained with $\gamma = 0$ until convergence and then trained for another phase with increasing λ values. We performed this experiment with dropout rate 0.2 and $\gamma = 2$, across 32 epochs.

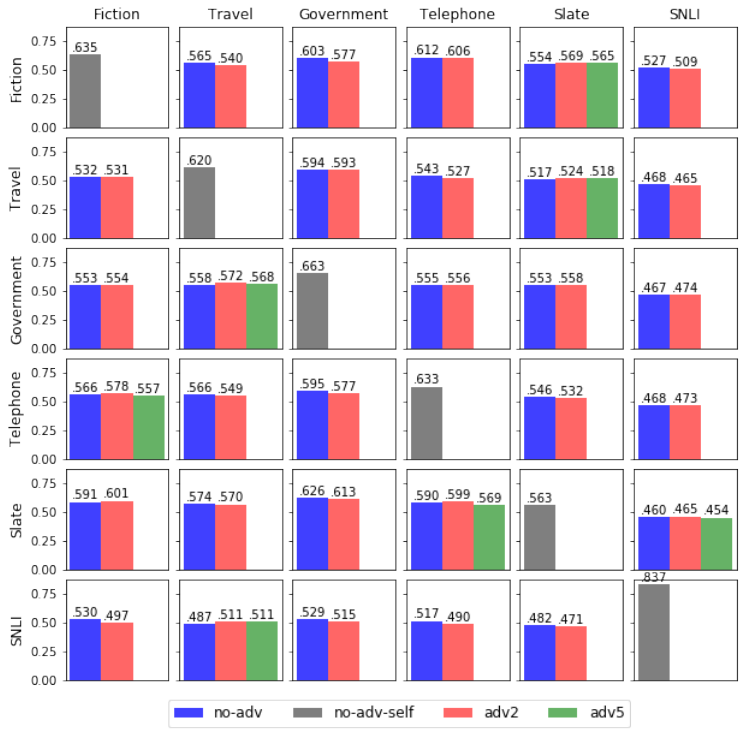

Figure 2: DANN2. Classification accuracy on the MULTINLI MATCHED and SNLI development sets, for different growth rates of λ. In the grid, rows are source domains and columns are target domains. Every cell contains between 1 to 3 models: NO-ADV or NO-ADV-SELF for $\gamma = 0$, ADV2 for $\gamma = 2$ and ADV5 for $\gamma = 5$.

4.3 Results

Results for the DANN2 baseline are summarized in Figure 2. Except SLATE, for every target genre the best performance was obtained when the same genre was the source domain (NO-ADV-SELF). Similar results were reported in [10]. In 13 out of the 30 domain combinations, training with $\gamma = 2$ (ADV2) obtained better accuracy than training with zero weight to the domain classifier (NO-ADV). Only in 7 of those cases the improvement was greater than 0.5%, where the greatest improvement of 2.4% was achieved in the SNLI-TRAVEL model. In all cases, increasing the growth rate to $\gamma = 5$ (ADV5) did not yield better performance.

Considering the results of the DANN5_1UL baseline, for most of the genres the accuracy decreases while increasing γ . There are cases, such as 9/11 and SLATE, in which there is a small improvement ($< 1\%$) when increasing γ from 2 to 5. Yet, the best accuracy is still obtained for $\gamma = 0$ (see Table 2 and Figure 3). Comparing the performance of this baseline to DANN2, a better accuracy is achieved when training on several domains. Concretely, for $\gamma = 0$ the averaged accuracy on the MATCHED genres in DANN5_1UL is 64.84%, in comparison to 62.29% in DANN2. Evaluation of DANN5_1UL on the test sets shows that the models generalize well to the MULTINLI MISMATCHED genres and generalize poorly to SNLI (see Table 3). Similar results were reported in [10] and in the leaderboards of the RepEval 2017 Kaggle's challenge[5].

Figure 4 shows the prediction accuracy across the training process of DANN5_1UL with 2ST. In the first training phase, when $\gamma = 0$, the accuracy increases until reaching to the early stopping point, on

[5] https://inclass.kaggle.com/c/multinli-matched-evaluation/leaderboard, https://inclass.kaggle.com/c/multinli-mismatched-evaluation/leaderboard

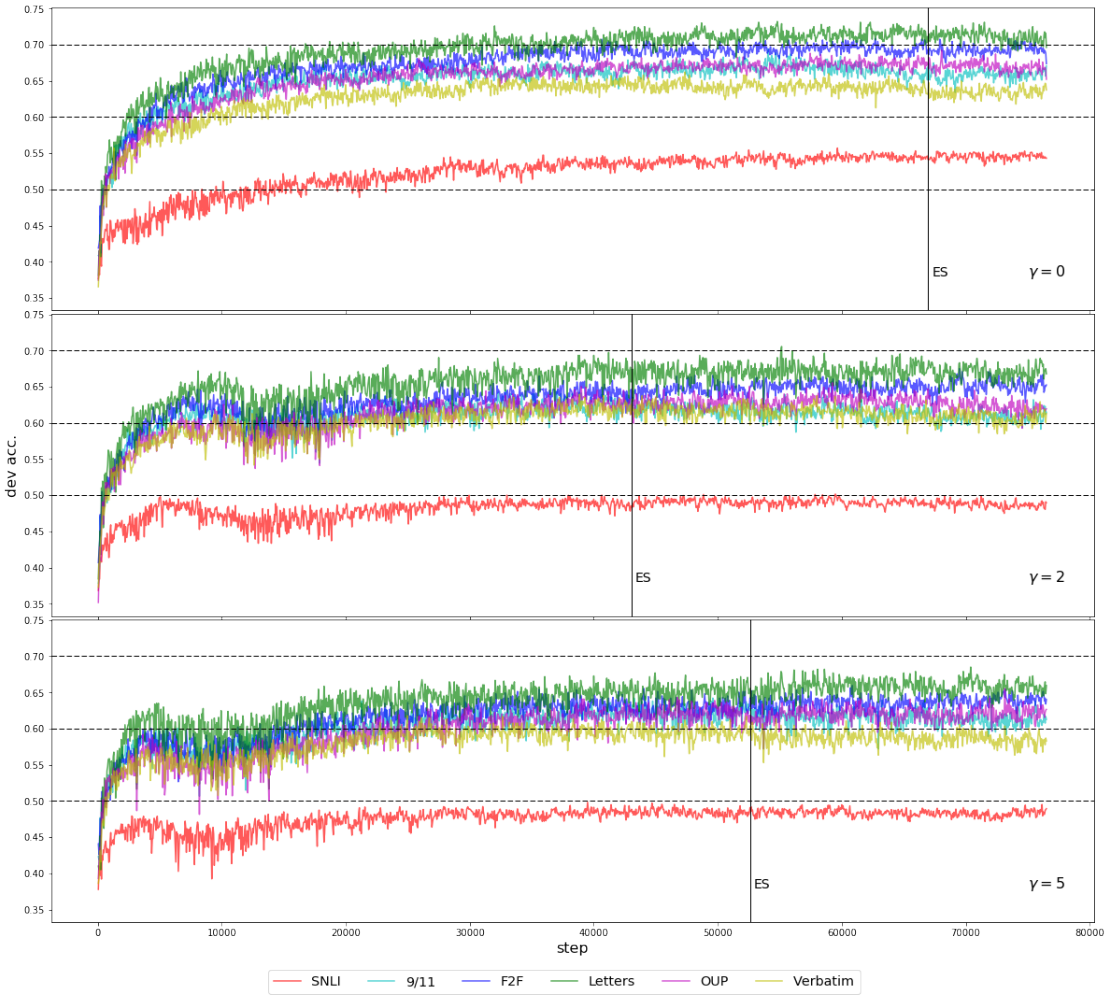

Figure 3: DANN5_1UL. Classification accuracy during training, on the MULTINLI MISMATCHED and the SNLI development sets, for different growth rates of λ. The 'ES' line marks the early stopping point.

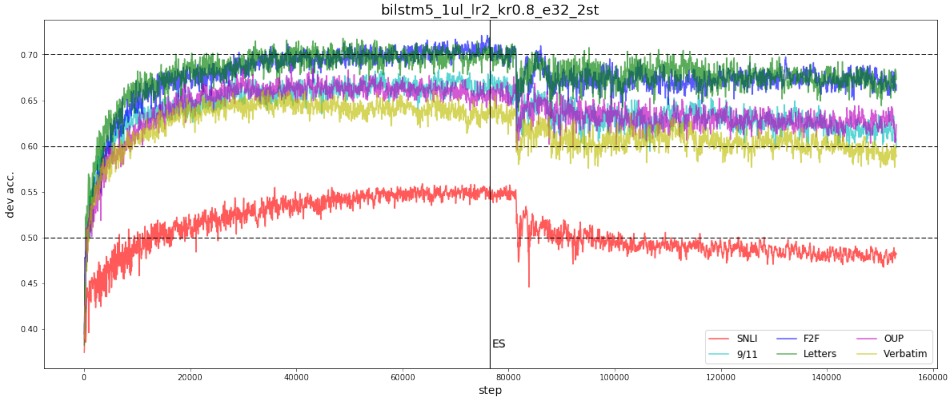

Figure 4: DANN5_1UL. Classification accuracy during two-step training, on the MULTINLI MISMATCHED and SNLI development sets. The model is trained with growth rate $\gamma = 0$ until the early stopping point (marked by the 'ES' line), on which it is set to $\gamma = 2$ for gradual increment of λ until the end of training.

	9/11	F2F	LETTERS	OUP	VERBATIM	SNLI
NO-ADV	0.6585	0.6994	0.7024	0.6677	0.6416	0.5572
ADV2	0.6167	0.6472	0.6667	0.643	0.62	0.5014
ADV5	0.6228	0.6319	0.6401	0.6204	0.5922	0.4983
	FICTION	TRAVEL	GOVERNMENT	TELEPHONE	SLATE	
NO-ADV	0.658	0.6305	0.6885	0.6665	0.5985	
ADV2	0.622	0.61	0.651	0.625	0.564	
ADV5	0.6175	0.594	0.636	0.627	0.5695	

Table 2: DANN5_1UL. Classification accuracy on the MULTINLI MATCHED and MISMATCHED genres and SNLI development sets, for different growth rates of : NO-ADV for $\gamma = 0$, ADV2 for $\gamma = 2$ and ADV5 for $\gamma = 5$. The results are based on the model parameters that yielded the best accuracy on SNLI development set during training

	MULTINLI MATCHED	MULTINLI MISMATCHED	SNLI
NO-ADV	0.6707	0.6751	0.5425
ADV2	0.6398	0.6408	0.4926

Table 3: DANN5_1UL. Classification accuracy on the MULTINLI MATCHED and MISMATCHED genres and SNLI test sets, of the NO-ADV and ADV2 selected models, as described in Table 2.

which the second phase starts. During the second phase, the weight of the domain classifier increases, while the accuracy is decreasing.

5 Analysis

To gain a better understanding of the results, we analyzed the MULTINLI dataset, and examined the performance of the domain classifier on the DANN5_1UL baseline. For data analysis, we randomly sampled 20 data points from each MULTINLI genre and from SNLI. We handcrafted 20 features, including content features (e.g. organizations and locations), grammatical properties (e.g. conjunction and explanation clause) and structural features (e.g. tense and 1st/3rd person). Based on these features, we constructed a vector representation for every sample, and performed a PCA analysis in two dimensions. The feature analysis is available at `https://tinyurl.com/y9na62kp`. The performance of the domain classifier was measured by its accuracy on the development set across training, for $\gamma \in 0, 2, 5$ [6].

5.1 DANN2

Generalization from the MULTINLI MATCHED genres to SNLI and vice versa is noticeably inferior to generalization from the MATCHED genres to both the MATCHED and MISMATCHED genres (Figure 2). In addition, when looking at the main diagonal in Figure 2 (NO-ADV-SELF), it is easy to see that when training on itself, SNLI achieves significantly superior results in comparison to those achieved by any of the MATCHED genres. These results can be explained by the notion that the SNLI dataset is degenerated in comparison to the MATCHED genres, as can be seen in Figure 5. The SNLI samples are highly condense as opposed to the MULTINLI genres. We believe that the low variance of the SNLI samples is the cause of the high accuracy that the SNLI archives on itself. Moreover, the density of SNLI makes it hard to generalize from SNLI to the MATCHED genres and vice versa, resulting in a low

[6] The measurements were performed on different executions, separately from the experiments described in 4

accuracy. One exception is the TRAVEL genre, which is also relatively condense (not as SNLI, but more than the other MATCHED genres). Accordingly, the best case where the adversarial training improved generalization, was obtained with SNLI as source domain and TRAVEL as target domain (see Figure 2).

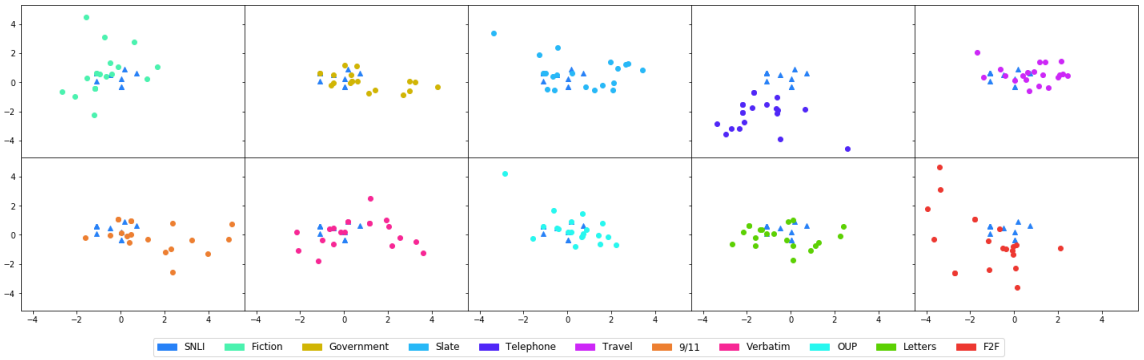

Figure 5: SNLI samples as blue triangles, compared to the samples of each of the ten MULTINLI genres, appears as colored circles. The samples are located on a 2D grid according to the first two dimensions of the PCA decomposition (based on all the samples).

Out of the 13 cases in which ADV2 obtained better accuracy than NO-ADV, 5 occurred on the GOVERNMENT row (see Figure 2). Namely, the adversarial approach improves generalization from this genre. Comparing the GOVERNMENT genre distribution to the distributions of the other MATCHED genres and SNLI, we can notice that besides FICTION and TELEPHONE, the majority of the target domain samples fall between the GOVERNMENT samples (see Figure 6). Moreover, the means of these distributions are placed among the GOVERNMENT samples. For these genres the improvement was more significant compared to FICTION and TELEPHONE.

5.2 DANN5

Examining the effect growth rate has on generalization, three clear trends can be observed (Figure 3):

First, as γ increases, the classification accuracy decreases. Considering the domain classifier performance, with zero weight ($\gamma = 0$) it reaches an average accuracy of $\sim 90\%$ on MULTINLI MATCHED and SNLI, in just 2 epochs (see upper subplot in Figure 7). This implies that the domain classification is an easier task in comparison to the label prediction, for which it takes more than 10 epochs to converge. When activating the domain classifier ($\gamma = 2, 5$), the average accuracy reaches a peak during the first epoch, after which it drops for 3-4 epochs and stabilizes around 20%-22% (see middle and bottom subplots in Figure 7). As there are 6 domains, the classifier performs almost as a random classifier.

Comparing the performance of the domain classifier and the label predictor during training, one can

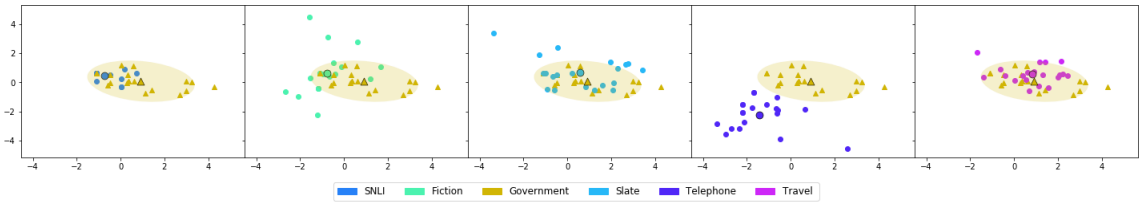

Figure 6: GOVERNMENT samples as yellow triangles, compared to the MATCHED genres and SNLI samples, which appear as colored circles. The mean of each genre is marked with a black outer-line. The ellipse mark the area which contains 95.4% of the GOVERNMENT distribution based on the 68-95-99.7 rule.

observe the correspondence between the accuracy drops of the two components. Concretely, during the first epochs there is a point at which the domain classifier reaches an accuracy peak, while the label predictor is still learning. At this point, the effect of domain classifier on the feature extractor increases, causing the performance of both the classifier and the predictor to decrease. Following this accuracy drop, the performance of the domain classifier stabilizes, while the performance of the label predictor slowly increases until reaching to a lower point in comparison to the model trained with $\gamma = 0$ (see Figures 3, 7).

The difference between the learning rates of the domain classifier and the label predictor is one possible explanation for the lower performance of the adversarial training. In addition, we suggest two other possible hypotheses: (1) Due to the similarity of the MULTINLI genres, when avoiding the adversarial approach, the model can exploit their common features, that may reveal the source domain without getting punished. This hypothesis is supported by highly similar performance of various models on the MATCHED and MISMATCHED genres. (2) By training the model on five different domains (DANN5), it is inherently being encouraged to extract non-domain-specific features, that will improve its prediction accuracy across all the training genres. In that case, the adversarial approach is becoming an additional regulator for the model.

A second observation is that SNLI converges to a significantly lower point than the MULTINLI MISMATCHED genres. This behavior could be explained by the argument of SNLI being degenerated in comparison to MULTINLI genres, as we discussed in section 5.1. This argument is also supported by the high accuracy of above 99% obtained by the domain classifier for this genre (see upper subplot in Figure 7).

Lastly, considering the graphs of $\gamma = 0, 2$, there is a noticeable difference between the convergence points of the MISMATCHED genres, which can be roughly divided into two groups. The first group contains FACE-TO-FACE and LETTERS, and the second group contains OUP, VERBATIM and 9/11. While the first group is characterized by semi/non-formal language and 1st/2nd person descriptions, the second group comprises of either informative or formal documents. Therefore, the different convergence points could be explained by the differences between the text genres.

5.3 DANN5 with 2ST

In 2ST, the domain classifier is activated only after the label predictor has converged. Hence, this setup is one way to address the unbalanced learning rates of the domain classifier and the label predictor, as described in the previous section.

Considering the label predictor, before the activation of the domain classifier, the model converges to a certain point for each genre. Once the first training step ends, the domain classifier is activated and there is a drop in accuracy (see Figure 4). This drop is expected, since until the second training step, the label predictor has the ability to exploit domain specific features. By increasing the weight of the domain classifier, such features are being suppressed. We believed that accuracy would increase again after this drop, as more inherent and cross-domain features will remain. Yet, the model does not recover and converges to a lower accuracy than at the end of the first training step.

Examining the domain classifier accuracy during training, a similar behavior as in DANN5 can be observed. The accuracy of both the classifier and the predictor drops after the ES point, where in this case the domain classifier convergences to a higher average accuracy of $\sim 28\%$ (see Figure 8). The hypotheses suggested in section 5.2 are given here as possible explanations as well.

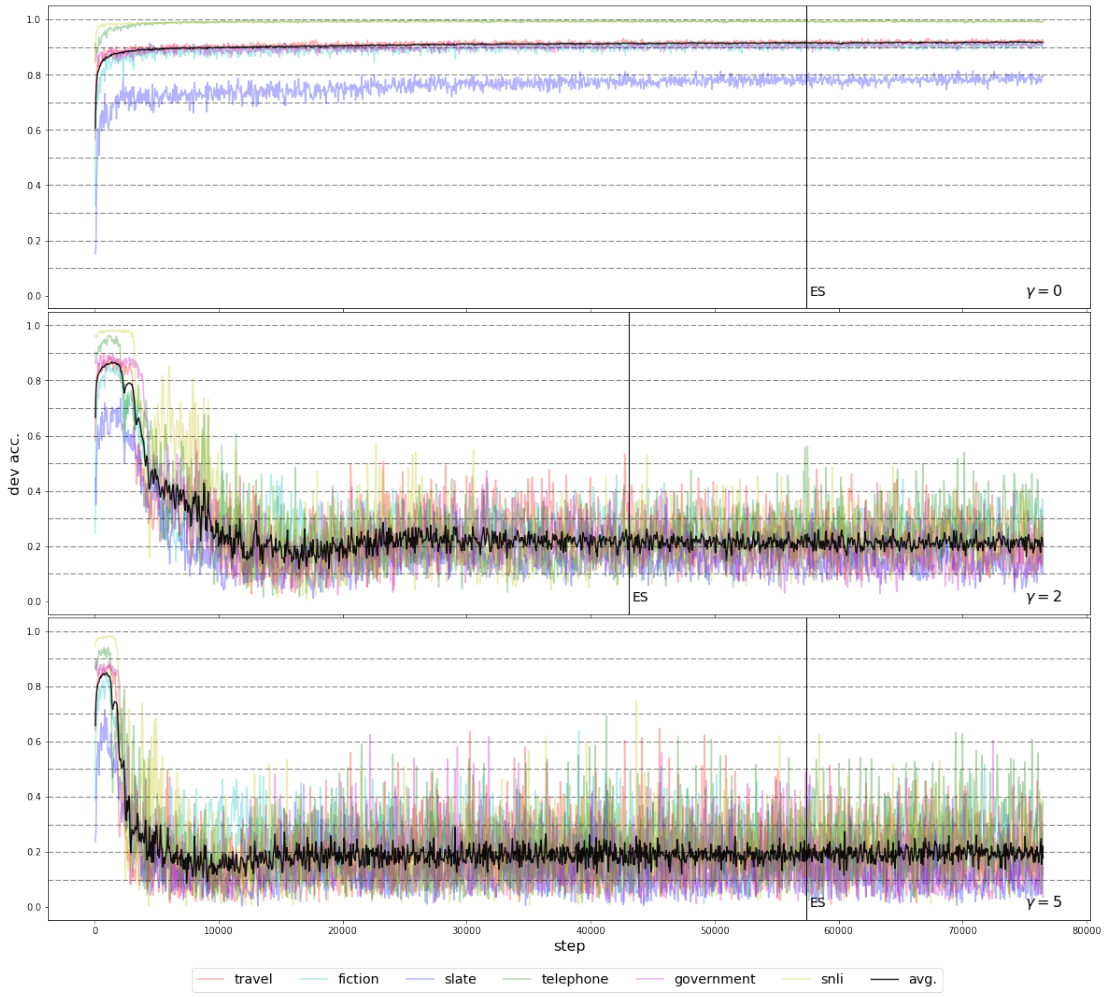

Figure 7: DANN5_1UL. Domain classification accuracy during training, on the MULTINLI MATCHED and SNLI development sets, for different growth rates of λ. The 'ES' line marks the early stopping point. The average accuracy is marked by the bold black line.

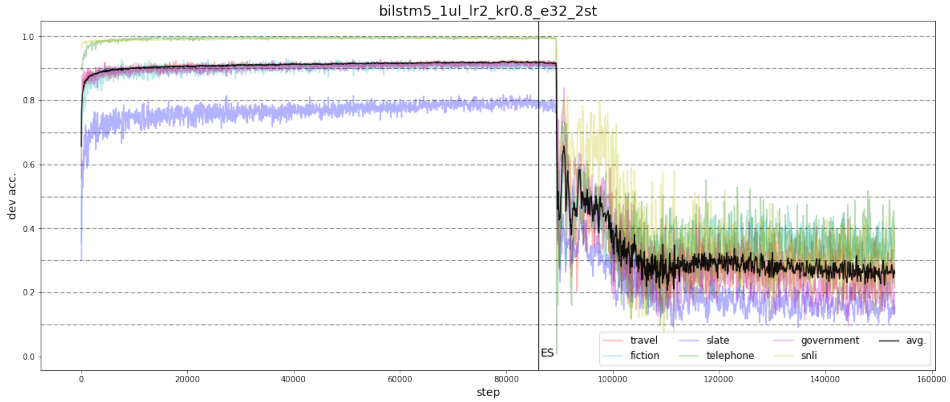

Figure 8: DANN5_1UL. Domain classification accuracy during two-step training, on the MULTINLI MATCHED and SNLI development sets. The model is trained with growth rate $\gamma = 0$ until the early stopping point (marked by the 'ES' line), on which it is set to $\gamma = 2$ for gradual increment of λ until the end of training. The average accuracy is marked by the bold black line.

6 Conclusions

In this work we applied a domain adversarial approach to the NLI task, using both the recently published MULTINLI dataset and the SNLI dataset. We evaluated two baselines: single source - single target (DANN2), and multiple sources - single target (DANN5). For 13 out of 30 domain combinations in DANN2, the adversarial approach improved the performance, although not significantly. A feature-based analysis of the data provided possible explanations for part of the results. In DANN5, the adversarial approach consistently reduced the performance. Here, one possible explanation could be the relatively high learning rate of the domain classifier in comparison to this of the label predictor. This hypothesis can be further examined either by decreasing the complexity of the domain classifier, or by using a more moderate growth function for its weight. Another two hypotheses are (1) the relatively high similarity of the MULTINLI genres, and (2) by training the model on multiple domains, it is inherently being encouraged to extract non-domain-specific features. In this case, the domain classifier is becoming an additional regulator for the model. As a future work, it would be interesting to put these hypotheses to test, by evaluating the extended setup of multiple source domains upon other tasks.

References

[1] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[2] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proc. ACL*, 2017.

[3] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[5] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*, 2015.

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[7] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649, 2015.

[8] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*, 2017.

[9] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[10] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[11] Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188*, 2017.