Using Unlabeled Out-Of-Domain Data to Improve Question Answering

Abstract

Our goal is to use unsupervised learning to improve the baseline of question-answering on the BioASQ dataset. We train our model on labeled training data from BioASQ, as well as unlabeled data from SQuAD (a significantly larger dataset), and we use a form of data augmentation by applying Gaussian noise to the word embedding input vectors to have the model obtain a more generalizable, domain-invariant underlying feature representation that is smooth over changes in the input space. Using this unsupervised loss term and optimizing only hyperparameters relevant to this loss term, we were able to get an approximate 6 point boost for EM and a 5.5 point boost for the F1 score on average on the BioASQ development set, although the models tend not to generalize very well to the unlabeled SQuAD examples used in unsupervised training, and can actually degrade performance on SQuAD if too much noise is used.

1 Introduction

When training a model on a small set of data, the model may have a tendency to overfit without developing features that encode a general understanding of the underlying data. Models trained on a domain from a single source tend not to generalize conclusions that are applicable to other sources (Desai et al., 2019).

Data augmentation artificially increases the amount of training data available without having to hand label more data, which is often expensive. This is especially hard to do in Natural Language settings because it is hard to perturb discrete words in a way that still preserves meaning (Zhang and Yang, 2018).

By leveraging additional sources of unlabeled data, and introducing a loss term that encourages the model to learn features invariant to noise in the unlabeled data, we hope to train a model that learns robust, domain-invariant features that ends up performing better on the dataset it was trained for than training without the proposed unsupervised loss term. By training on the unlabeled dataset, we will get features that are smooth over changes in the input of the unlabeled dataset domain, meaning that the features should overfit less. Ideally, leveraging this larger source of additional data (despite not using the labels) will give the model the training signal of a much larger dataset, which will lead to significantly improved results.

We use a loss term as described in (Xie et al., 2020), which augments an example of unlabeled data, runs both the data and perturbed data through the model, and minimizes a divergence metric (as L1 or MSE Loss) between the model's predictions for the data and perturbed data. This loss term therefore encourages the model to learn features that is not sensitive to applying noise to the data, which will hopefully be more robust.

While adjusting discrete words and preserving meaning is difficult, especially in the question-answering setting where we need to maintain a direct relationship between the question and span texts, we can instead perturb unlabeled data by applying noise to the continuous word embeddings generated by the data. We used this idea to create our loss term and train our model.

2 Approach

2.1 Data Sources

Because we only had one labeled dataset for BioASQ, we decided to split this dataset into a training set as well as a test set, with 60% of the overall data as training, and 40% as the test set. For the unsupervised learning, we used the training data of the SQuAD dataset, but without the labels. Our overall training data thus consists of both labeled data from BioASQ, as well as unlabeled data from SQuAD.

2.2 Unsupervised Loss Term

After obtaining the embeddings of both the question and passage, we add on some Gaussian noise, scaled by a hyperparameter constant. To obtain the unsupervised loss term, we run our model on both the normal embeddings as well as the perturbed embeddings, and run some divergence metric on the difference between the two. Specifically, representing the model as f, the original input as x, the perturbed input as x', and the divergence metric as d, we add on $d(f(x), f(x'))$ as the unsupervised loss term. The overall loss is the sum of the supervised loss and the unsupervised loss, where the unsupervised loss is scaled by a hyperparameter.

2.3 Hyperparameters

We varied the weight of this loss term (how much the unsupervised loss contributes to the overall loss), the scaling constant of the Gaussian noise we applied to the embedding vectors (how large our perturbation to embedding vectors are), and the specific divergence metric (L1 vs MSE Loss) used to calculate the unsupervised loss term.

2.4 Experiments

We initially trained the baseline model with the default settings of the given starter code to obtain a baseline to compare with our results. The default parameters included a hidden dimension of size 128 with a bidirectional LSTM. This baseline model was trained on the BioASQ training set.

After implementing the unsupervised loss, we did a grid search over a variety of different hyperparameter values to try to determine optimal settings for the model relating to the unsupervised loss term, and attempted to measure the improvement between the models trained using the loss term and the baseline model with no unsupervised loss term. Specifically, we trained models weighting the unsupervised loss term at $0.05, 0.5, 5.0$, scaling the Gaussian noise by $0.1, 1.0, 10.0$, and trying L1 Loss versus MSE Loss as the divergence metric for the unsupervised loss term. We trained 3 models for each of the 18 possible settings of each of the hyperparameters and evaluated the results on our generated BioASQ Development dataset and the SQuAD Development dataset (to see if the model could learn to generalize to the SQuAD dataset without explicitly seeing any SQuAD labels). In our results setting, we present the average EM and F1 of the models trained, with the specific results for each model run in the appendix.

We ended up running a few auxiliary experiments to further explain the results we saw (specifically running training but reusing the training set examples without labels for the unsupervised loss term instead of a disjoint domain dataset), which are further explained in the following section.

3 Results

First, to establish a baseline, we ran the default training for the baseline model with a hidden size of 128 and had an average EM of 54.82 and an average F1 of 63.17.

Here, the results using MSE loss are shown in Tables 1 and 2. The results using L1 loss are shown in Tables 3 and 4.

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	54.99	55.26	56.75
1.0	57.59	59.36	60.85
10.0	57.14	59.13	56.81

Table 1: Average EM on BioASQ using MSE Loss

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	63.50	63.54	64.98
1.0	65.35	67.54	68.68
10.0	64.96	67.26	64.23

Table 2: Average F1 on BioASQ using MSE Loss

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	51.55	52.88	54.04
1.0	57.81	60.19	59.35
10.0	56.98	56.20	54.98

Table 3: Average EM on BioASQ using L1 Loss

In general, having a noise magnitude of 0.1 seems to have a lower EM and F1 than noise magnitudes 1.0 and 10.0. The values for noise magnitude of 0.1 are also close to the original base line of EM 54.82 and F1 of 63.17, which suggests that

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	59.90	60.85	63.42
1.0	64.74	67.71	68.02
10.0	65.54	63.44	62.94

Table 4: Average F1 on BioASQ using L1 Loss

a very tiny change in our embeddings for the unsupervised loss term does not change the results very much.

In most cases, having a noise magnitude of 1.0 seems to out-perform the noise magnitude of 10.0 in both EM and F1 as well, though not as much of a change compared to noise magnitude 0.1. We initially predicted that having a high noise magnitude of 10.0 would lead to significantly worse performance because applying so much noise would likely significantly alter the embeddings and would cause them to map to an example with a completely different meaning. However, it seems that the performance on the BioASQ Development dataset did not degrade nearly as much as expected, and the smoothness over the noisy SQuAD examples helped the model to learn a better representation for BioASQ question answering, on average.

For MSE Loss and noise magnitude 0.1 or 1.0, having loss weight as 5.0 seemed to have the best performance. On the other hand, for noise magnitude 10.0, having loss weight as 1.0 seemed to have the best performance. As for L1 Loss, the best performance alternates between loss weight 0.5 and 5.0. On average, the models trained with the unsupervised loss term calculated with MSE Loss did slightly better than the models trained with the unsupervised loss term calculated with L1 Loss, although the models trained with the best combination of hyperparameters but different divergence metrics were still comparable (both were able to reach EM $\approx 59-60$ with F1 ≈ 68).

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	2.65	2.46	1.26
1.0	0.97	0.60	0.71
10.0	0.63	0.51	0.58

Table 5: Average EM on SQuAD using MSE Loss

In addition, we wanted to see how our model, trained with labels from BioASQ, would perform

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	8.02	7.63	4.55
1.0	5.43	4.02	4.84
10.0	5.34	3.96	4.25

Table 6: Average F1 on SQuAD using MSE Loss

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	2.69	2.54	1.46
1.0	1.68	0.76	0.69
10.0	1.09	0.57	0.62

Table 7: Average EM on SQuAD using L1 Loss

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	7.81	8.04	6.13
1.0	7.19	4.97	4.95
10.0	6.28	4.20	4.01

Table 8: Average F1 on SQuAD using L1 Loss

on the SQuAD development set, even though our model was not trained on any labeled SQuAD data. Our results are in Tables 5,6,7, and 8. We performed three trials over the same hyperparameters as before (noise magnitude, loss weight, and type of loss). Overall, the average EM and F1 values are low. This is understandable, since our model has never seen any labels for SQuAD, but it was unfortunate that the model's performance was still so poor. Initially, we thought that it either did not learn features that are fully generalizable to the SQuAD dataset or did not have its weights tuned to the specific domain. We believed that the poor performance is a result of being unable to deal with unseen words not evident in the BioASQ dataset, because the unsupervised loss term doesn't provide enough of a signal to figure out what to do with these unknown words without access to the unsupervised dataset labels.

However, upon further analysis, we came to some new conclusions. Interestingly, the trained model performance paradoxically degraded on the SQuAD set the higher that we set the loss weight term, which is strange because placing more emphasis on these training examples should lead to more focus and better performance for the unsupervised data. We believed that this was likely because the noise that we applied to the SQuAD data was large enough in magnitude that the perturbed embeddings actually represented an example with significantly different meaning than the example without the perturbation. Because the model isn't incentivized to get the SQuAD answers correct (as it does not see the labels), it does not generalize well to the SQuAD dataset. In fact, it does worse because it learns to treat SQuAD examples with different meaning identically by seeking to minimize the divergence between the original and perturbed examples.

To test this hypothesis, we used unsupervised training data from BioASQ (as opposed to unsupervised training data from SQuAD), reusing our training data to calculate the unsupervised loss term. Ideally, if our hypothesis was correct, we would see a much poorer performance in the BioASQ development set as the loss term and noise magnitude increased. As expected by the hypothesis, the results suggest that the model performs well when noise magnitude is about 0.1, while it performs significantly worse than the baseline when the noise magnitudes are 1.0 and 10.0 (with average EM dropping to mid 40's). See Tables 9, 10, 11, and 12.

It seems that applying only a small perturbation (small noise magnitude) helps the model be invariant to small changes, while using large changes within the same dataset forces the model to see very different examples as being the same, resulting in poor performance. Although we don't see a domain adaptation benefit to using a different domain for the unsupervised loss term, it does help the model to improve without being so sensitive to tuning this loss term because higher magnitudes of the loss term and noise magnitude will mainly degrade performance on the unsupervised set, rather than the specific set that the model is training on, as evidenced by our data.

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	55.65	58.47	56.64
1.0	55.81	59.63	45.18
10.0	56.98	57.64	43.69

Table 9: Average EM on BioASQ using MSE Loss, unsupervised loss term on BioASQ

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	63.62	65.55	65.62
1.0	63.47	66.22	54.88
10.0	64.20	63.84	53.38

Table 10: Average F1 on BioASQ using MSE Loss, unsupervised loss term on BioASQ

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	56.31	61.79	62.79
1.0	50.5	51.83	43.02
10.0	53.65	55.81	42.19

Table 11: Average EM on BioASQ using L1 Loss, unsupervised loss term on BioASQ

Noise	**Loss Weight**		
Magnitude	0.05	0.5	5.0
0.1	63.93	69.85	69.53
1.0	59.62	61.35	53.72
10.0	63.4	62.62	51.73

Table 12: Average F1 on BioASQ using L1 Loss, unsupervised loss term on BioASQ

Throughout our data, our best performing model on the BioASQ development dataset was trained with a loss weight of 0.5 and a noise magnitude of 1.0, with an EM of 66.61 and a F1 Score of 73.02. Unfortunately, this model was overwritten by further tests, but we were able to train a similar performing model (using a loss weight of 5.0 and a noise magnitude of 1.0) with an EM of 64.62 and an F1 Score of 71.88, which we have included in our submission as *best_model.pt*, which gives an improvement of about 10 points of EM and 8.5 points of F1 compared to the average baseline. Given that we're cherry picking the best result based on our development set optimizations, it's likely that this model overfits to our choice of development set, but we thought it'd be good to include the best model that we trained to give an idea of the explored techniques' full potential.

4 Conclusions

Overall, the explored technique seems to provide concrete advantages for the Question Answering task. We were able to see an average EM increase of about 6 points and an average F1 increase of about 5.515 points given our best setting of hyperparameters involving the unsupervised loss term and keeping everything else constant. We think that this term helps the model because it helps the model to learn smoother features that are more invariant to noise overall.

However, while seeing the SQuAD dataset seems to help the model learn a good feature set that is invariant to noise in the SQuAD domain, it does not seem to improve performance on the dataset where labels are not provided, and can actually degrade performance if the noise magnitude is set to a high constant. We believe that it is likely because while the model has a feature set that is invariant to noise in the SQuAD domain, because it does not see the SQuAD data labels and does not explicitly train to labels (just the canonical representation), it does not learn how to best use the features to make predictions in the SQuAD dataset because the weights are only explicitly tuned for BioASQ. For higher noise magnitudes, the model learns to treat perturbed examples of the SQuAD dataset identically to the original example, when in reality this assumption degrades performance and leads to the trends evidenced in our data. An interesting future extension might be using transfer learning with different datasets to see if the smooth features encouraged by the unsupervised loss term translate to success in different domains.

Acknowledgments

We'd like to thank Dr. Durrett and the TAs for running such a great and helpful NLP class!

References

Shrey Desai, Barea Sinno, Alex Rosenfeld, and Junyi Jessy Li. 2019. Adaptive ensembling: Unsupervised domain adaptation for political document analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 4718–4730.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised data augmentation for consistency training. *ArXiv*, arXiv:1904.12848v6.

Dongxu Zhang and Zhichao Yang. 2018. Word embedding perturbation for sentence classification. *ArXiv*, abs/1804.08166.