Is it better to teach yourself or to be taught? A comparison of self-learning versus active learning for improving NLU robustness

Pooja Sethi[*]
Department of Computer Science.
Stanford University
pjasethi@stanford.edu

Abstract

Self-learning has shown to be an effective way of improving the robustness of NLU models. Similarly, active learning has shown to be an effective method of improving sample efficiency and adapting to changes over time (concept drift). However, the two paradigms seem to be fundamentally at odds. The first augments the training set by sampling the *most confident* utterances and using the model *predictions* as proxy-labels, while the latter samples the *least confident* utterances, for which *gold annotations* are obtained. In this work, I empirically compare the effectiveness of self-learning and active learning for improving both binary and multi-class text classification accuracy, using the CivilComments and Amazon datasets from the WILDS benchmark, respectively. I find that a hybrid approach combining active learning and self-learning exhibits the best performance on CivilComments and that active learning alone exhibits the best performance on Amazon. Interestingly, self-learning alone provides slim improvements in one case and degrades performance in the other. These findings indicate that the best choice of data augmentation strategy is dataset dependent. Code, trained models, and experiment results are available at this https url.

1 Introduction

Text classification is a hallmark task in natural language processing (NLP) and understanding (NLU). Classification models are essential to many real-world applications. For example, such models can help predict user sentiment on product reviews (*"Did you love this book or hate it?"*); route a user's request to a virtual assistant to the appropriate domain (*"Did you ask Alexa for the weather or to play music?"*); or automatically organize documents into categories (*"Did you upload an invoice, receipt, or legal document?"*).

An NLU model is *robust* if, when trained on source distribution d_s, it is able to transfer well on a related but different target test distribution d_t. There are several ways to characterize this domain shift, as discussed by Koh et al. (2021):

Domain generalization: d_s is disjoint from d_t. For example, a sentiment classifier is trained on IMDB movie reviews but then applied to Amazon book reviews.

Subpopulation shift: d_s and d_t have overlapping but different proportions of a feature or class. For example, a language identification model is trained in the US with a small proportion of Urdu

[*]This work was done as an independent research project for CS 329D. The author is jointly affiliated with and kindly supported by Impira Inc. Contact information: pooja@impira.com

examples, but then applied in South Asia (where the relative proportion of Urdu speakers is much higher).

Hybrid: d_s and d_t exhibit domain generalization across one feature and subpopulation shift across a different feature or class. For example, a speech recognition model may need to be robust to entirely new dialects, e.g., Australian-English in addition to American-English (domain generalization). Over time, it may also need to learn the vernacular of an increasingly young userbase (subpopulation shift).

Note that in this work, we assume that the set of class labels is held constant between d_s and d_t, i.e., zero-shot transfer is a non-goal.

A variety of semi-supervised approaches have emerged for domain adaptation in NLP, including pre-training and self-learning. These approaches attempt to use *unlabeled* data in a meaningful way to improve the performance on d_t. In real applications, however, assuming only access to unlabeled data may be an overly pessimistic constraint. Although the budget may be limited, it is often possible to annotate additional examples from the same or similar distribution as d_t. Indeed, active learning is a commonly used method that selectively incorporates additional labeled data into the training set.

The objective of this work is to evaluate and compare data augmentation methods for NLU domain adaptation, particularly self-learning, a semi-supervised approach (teaching oneself), with active learning, a supervised approach (being taught). In particular, I answer the following questions:

- How does adding additional training data via self-learning compare to adding the equivalent amount of data via active learning?
- What happens if traditional self-learning is combined with traditional active learning?
- What happens if self-learning is done with low-confidence examples? What happens if active learning is done with high-confidence examples?

In Section 2, I discuss related investigations. In Sections 3 and 4, I expand on the sampling methods tested, the experiment setup, and the empirical findings. Finally, in Section 5, I reflect on the results and discuss opportunities for future work.

2 Related Work

While there has been a volume of work on self-learning and active learning for NLP individually, to my knowledge, there hasn't been a direct comparison of these approaches.

Semi-supervised learning, and specifically self-training, has shown promising results in various NLP tasks. Ruder & Plank (2018) find that self-training "achieves surprisingly good results" on sentiment classification in the context of domain shifts. They also find that tri-training, which can be seen as an extension of the more simple self-learning idea, even out-performs neural domain adaptation methods (Ganin et al., 2016; Saito et al., 2017). Du et al. (2020) show that self-training improves pre-training for NLU, with up to 2.6% improvements on standard text classification benchmarks.

Active learning has also been widely used for improving NLU models (Duong et al., 2018; Peshterliev et al., 2019; Sen & Yilmaz, 2020). Though the specific sampling methods used vary across the literature, many are uncertainty-score based, as described by (Monarch, 2021). A simple least-confidence strategy was shown to be surprisingly effective for improving bug detection (which implicitly could lead to robustness to domain shifts) (Sethi et al., 2021).

Unfortunately, the literature seems to be more sparse when it comes to comparisons of self-learning and active learning. (Ramponi & Plank, 2020) survey approaches to domain adaptation for NLP, but their survey focuses more heavily on neural unsupervised approaches as opposed to data-driven approaches to domain adaptation.

3 Task

The task setup includes a text classification model as well as a library of sampling strategies that are used to perform augmentation on the base classifier.

3.1 Natural Language Understanding

A prototypical NLU task is text classification. Given a sentence (or multi-sentence paragraph), the objective is to predict the class it belongs to. More formally, we can assume the input to this model is a string of text x_i and label y_i. The model predicts a label for this text, $\hat{y}_i$. It may also return a distribution of confidence scores among C classes where c_i^j is the model's confidence that $\hat{y}_i = C^j$. Note that the confidence scores may not necessarily be probabilities, since the models used for text classification are often uncalibrated.

For our evaluation, we assume that we have a seed text classification model parameterized by $f; \theta_0$ trained on N training examples. The task is then to improve the seed model using iterated rounds of data augmentation.

3.2 Sampling Approach

The approach to sampling is given by Algorithm 1.

Algorithm 1 Data sampling and retraining procedure

```
Input:
  $f; \theta_0$ An initial model trained on the full training set $X$
  $d$ A decision function that returns confidence scores for $f$
  $U$ An unlabeled candidate pool of utterances
  $U_{gold}$ Gold annotations for the unlabeled candidate pool, assumed to be given by some oracle.
  $S$ A set of sampling strategies (functions)
  $k$ A number to sample per strategy
  $n$ The number of sampling rounds

  procedure SampleAndTrain($f, d, \theta_0, U, S, k, n$)
    for $i \leftarrow 1$ and $i <= n$ and $i++$ do
      for $s \in S$ do
        $U_{predictions} \leftarrow []$                    ▷ Save the predictions on the unlabeled data
        $U_{scores} \leftarrow []$                         ▷ Save the confidence scores on the unlabeled data
        for $u_j \in U$ do
          $\hat{y_{u_j}} = f(u_j, \theta^s_{n-1})$         ▷ Get predictions using model with previous parameters
          $U_{predictions}[j] \leftarrow \hat{y_{u_j}}$
          $c_{u_j} = d(u_j, \theta^s_{n-1}, s)$            ▷ Get confidence scores using decision function
          $U_{scores}[j] \leftarrow c_{u_j}$
        end for
        $selected \leftarrow s(U_{predictions}, U_{gold}, U_{scores}, k)$    ▷ Select $k$ samples using $s$
        $\theta^s_n \leftarrow \theta^s_{n-1}$ incrementally trained with $selected$ examples.
      end for
    end for
  end procedure
```

I assume access to a candidate pool of U unlabeled examples and that in total we'll perform n round of sampling. At each round, model predictions and confidence scores are obtained using the models checkpointed from the previous iteration. (There will be a previous model unique to *each* sampling strategy s, except for in the first iteration, in which θ_0 is shared.) The confidence scores are then used by the various sampling strategies to determine which k examples to select from U. For each strategy, a new model is trained using its selected samples.

3.3 Sampling Strategies

In total, seven sampling strategies are tested and compared. The approach taken by each sampling method is discussed in the subsequent subsections.

3.3.1 Random (Baseline)

Select and return k utterances randomly from U and use their **gold annotations**.

3.3.2 Self-Learning

Sort U by descending order of confidence scores c_i (higher confidence is at the top). Select and return the **top** k utterances and the corresponding **predictions** $\hat{y}$ from the sorted list.

3.3.3 Active Learning (Uncertainty Sampling)

Sort U by descending order of confidence scores c_i. Select and return the **bottom** k utterances and the corresponding **gold annotations** y from the sorted list. We assume access to an oracle for obtaining the gold annotations.

3.3.4 Hybrid

Sort U by descending order of confidence scores c_i. Next, combine the two methods above:

1. Select the **top** $k/2$ utterances and the corresponding **predictions** $\hat{y}$ from the sorted list.
2. Select the **bottom** $k/2$ utterances and the corresponding **gold annotations** y from the sorted list.

Merge the top and bottom results together to return k samples. The intuition behind this method is that it may be beneficial to both have the model reinforce what it already knows (self-learning) and have an oracle guide it on the areas where it's not confident (active learning).

3.3.5 High-Confidence Use Gold

This method does active learning as in Section 3.3.3, but returns the **top** k high confidence examples. Intuition suggests that this method should perform better than self-learning (because we obtain oracle labels), but it is unclear whether it would be better than active learning.

3.3.6 Low-Confidence Use Prediction

This method does self-learning as in Section 3.3.2, but with the **bottom** k low confidence examples. Intuition suggests that this method should perform poorly, especially since the model may have higher likelihood of being incorrect when the confidence score is low.

3.3.7 Low-Confidence Inverse Prediction

This method does self-learning as in Section 3.3.2, but with low confidence examples and uses the **opposite** of the predicted class (for binary classification) or the second-highest predicted class (for multi-class classification) as the label. The intuition behind this method is that if the model has low confidence on its prediction, it may be better to use its next-best guess as the label.

4 Experiments

4.1 Datasets

Evaluation was conducted on two NLP (classification) datasets from WILDS (Koh et al., 2021).

CivilComments This is a binary classification task to determine whether a given sentence or paragraph is toxic (Borkan et al., 2019). In addition to the input text x and the binary toxic label y, the dataset also contains metadata for each utterance containing 8 possible demographic identities, or subgroups, that the utterance could belong to (*male*, *female*, *LGBTQ*, *Christian*, *Muslim*, *other religions*, *Black*, and *White*). In addition to overall test accuracy, the worst-case subgroup accuracy is also reported. The goal is to avoid bias towards any particular subgroup when predicting toxicity while also maintaining high overall accuracy.

Amazon This is a multi-class classification task to predict the star rating, on a scale of 1-5, given with an Amazon product review (Ni et al., 2019). In addition to the input text x and the rating label y, the dataset also contains metadata for each utterance containing the ID of the person who wrote the review, referred to as the *domain*. In addition to overall test accuracy, the tail performance on different subpopulations of reviewers is also reported. The goal is to achieve high tail performance while also maintaining high overall accuracy.

4.2 Model

The model used for the experiments consists of two main components.

SBERT Sentence Encoder First, the input sentence x_i is embedded into 384-dimensional vector using a pre-trained Siamese BERT network (Reimers & Gurevych, 2019). This model is specifically trained to compute sentence embeddings that can be compared with cosine-similarity, and it's trained on more than 1 billion training pairs. The particular pre-trained model chosen for these experiments was `all-MiniLM-L6-v2`. I chose SBERT in particular because it achieves state-of-the-art performance over other sentence embeddings (such as GloVe (Pennington et al., 2014), InferSent (Conneau et al., 2018), and Universal Sentence Encoder (Cer et al., 2018) while being computationally efficient.

Linear SVM with SGD The actual classification task is done using a linear SVM model. More specifically, using the `SGDClassifier` from `sklearn` with hinge loss (Pedregosa et al., 2011), which enables scaling to a large number of samples. The data given to the SVM is the embedded input e_{x_i} (which comes from the SBERT model) and the class label y_i.

Because the primary objective of these experiments is to evaluate and compare data augmentation methods as opposed to improve the underlying model itself, the model architecture I use is deliberately simple and designed to enable fast iteration. In total, 70 models were trained (7 sampling methods x 5 iterations x 2 datasets) on a CPU.

4.2.1 Confidence Score Calculation

It is worth noting how I compute confidence for the SVM model. For the binary classification case, the `decision_function` of `SGDClassifier` returns a score corresponding to its confidence in the prediction of the positive class. If the score is > 0, the positive class is predicted. Otherwise, the negative class is predicted. The signed score is proportional to the distance of the point to the learned hyperplane. Thus, I assign `confidence = abs(score)`.

For the multi-class case, the `decision_function` of `SGDClassifier` returns an array of scores, where each index contains the confidence of the corresponding class. The predicted class is the `arg max` of this array. Thus, I assign `confidence = max(scores)` .

Note that the confidence is *not* equal to a calibrated probability. While there is an option to calibrate the confidence using Platt scaling (Platt, 1999), it is noted in the documentation of `SVC` that it slows down training and the `arg max` of the probabilities may be inconsistent with the `arg max` of the scores. Since a calibrated probability is neither a necessity for active learning nor self-learning I forgo this option.

4.2.2 Online Learning

For the "seed" round of training ($i = 0$), a model is trained on the full training set using the `fit` function of the `SGDClassifier`. For all subsequent rounds ($1 \leq i \leq 5$) the checkpointed model from the $i = (n-1)^{th}$ round is loaded and updated in an online fashion using the `partial_fit` function. The data that is used for the online rounds comes from the full validation set. In each round, the entire validation set is considered and 10% of it is selected per the relevant sampling strategy.

4.2.3 Re-weighting

Finally, in order to make sure minority groups were given appropriate weight, I also applied re-weighting in the seed round ($i = 0$) of training. The weights for each group are automatically calculated using `compute_class_weight` using the inverse frequency of the classes in the training data. For CivilComments, the computed class weights were approximately `{0: 0.56, 1: 4.40}` (toxic labels are more rare and heavily up-weighted) and for Amazon they were approximately `{0:`

Table 1: CivilComments Test Accuracy (and Worst-Group Accuracy for $n = 5$)

	Sampling Round					
Method	0	1	2	3	4	5
Random	0.748	0.739	0.763	0.751	0.771	0.745 / 0.274
Self Learning	0.748	0.748	0.748	0.749	0.749	0.750 / 0.302
Active Learning	0.748	0.858	0.677	0.854	0.664	0.854 / **0.548**
Hybrid	0.748	0.874	0.737	0.875	0.722	**0.867** / 0.486
High Confidence Use Gold	0.748	0.837	0.807	0.799	0.801	0.801 / 0.436
Low Confidence Use Prediction	0.748	0.485	0.227	0.127	0.114	0.114 / 0.0
Low Confidence Inverse Prediction	0.748	0.415	0.165	0.115	0.473	0.182 / 0.002

`18.54, 1: 7.28, 2: 2.14, 3: 0.68, 4: 0.34}` (lower star ratings are more rare and heavily up-weighted).

4.3 Results

Results on both the CivilComments and Amazon dataset across repeated rounds of sampling and retraining are shown in Figure 1. The raw average accuracies for both datasets are shown in Tables 1 and 2, respectively.

4.3.1 Average Test Accuracy

A hybrid approach (combining active learning and self-learning), or active learning alone, does better than self-learning alone. In the case of Civil Comments, hybrid does the best, achieving 0.867 test accuracy and in the case of Amazon, active learning does the best with 0.527 test accuracy. However, these results can't be taken at face-value. For CivilComments, although the hybrid and active learning alone approaches do significantly better than all other methods in round 5, they are unstable (i.e., they "zig zag") and in prior iterations sometimes perform worse than other methods. For Amazon, the curve for hybrid and active learning is much more smooth across iterations. However, active learning (the best method on this dataset) does only marginally better than random sampling, which itself only marginally improves over the seed model. This indicates that there may be limits to how much data augmentation can help.

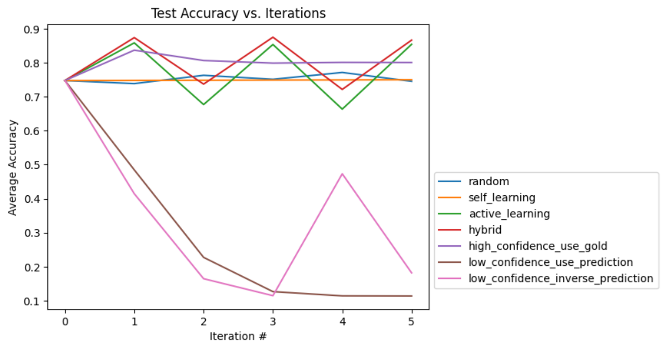
The results for self-learning alone are underwhelming. For Civil Comments alone, self-learning does only marginally better than random sampling. On Amazon, self-learning actually *hurts* performance. Finally, among the rest of the methods high-confidence use gold does well, often significantly better than self-learning. This indicates that active learning on high-confidence examples may also be a worthwhile endeavor.

4.3.2 Worst Group Accuracy

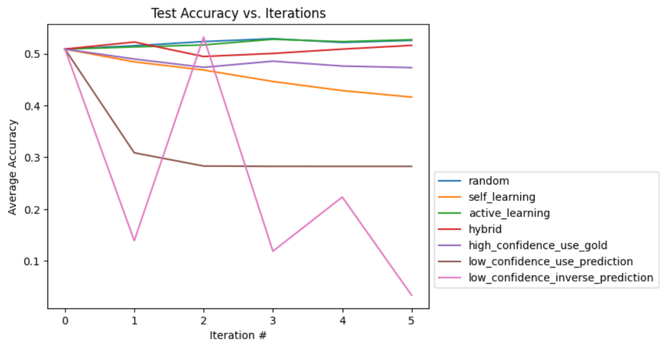
The worst-group accuracies for both datasets are shown for $n = 5$ in Tables 1 and 2. Active learning achieves the best worst-group accuracy on CivilComments, which is perhaps unsurprising since active learning specifically targets uncertain examples which are likely to correlate with underrepresented groups. The results on Amazon are more surprising. Random sampling achieves the best worst-group performance by a significant margin. This could indicate that active learning is systematically excluding some minority groups, perhaps because they are low confidence but don't have the *lowest* confidence.

5 Discussion

Limitations It's worthwhile to note a few limitations of this work. First, it is expected that the performance of the best models here will be within the range of but not necessarily competitive with those of the WILDS leaderboard (Koh et al., 2021). The overall objective of this work was to compare data-centric approaches to domain adaptation by first training an initial model on the provided training set and then doing additional data augmentation using sub-samples of the validation



(a) CivilComments



(b) Amazon

Figure 1: Accuracy on the test set versus the sampling round. All the sampling methods start from the same seed model. They diverge as additional data is sampled and used to update the model from the previous iteration in an online fashion.

Table 2: Amazon Test Accuracy (and Worst-Group Accuracy for $n = 5$)

	Sampling Round					
Method	0	1	2	3	4	5
Random	0.509	0.516	0.524	0.529	0.522	0.526 / **0.053**
Self Learning	0.509	0.484	0.469	0.446	0.429	0.416 / 0.013
Active Learning	0.509	0.513	0.517	0.528	0.523	**0.527** / 0.027
Hybrid	0.509	0.523	0.495	0.501	0.509	0.516 / 0.027
Low Confidence Inverse Prediction	0.509	0.139	0.532	0.118	0.223	0.034 / 0.0
High Confidence Use Gold	0.509	0.49	0.474	0.486	0.476	0.473 / 0.013
Low Confidence Use Prediction	0.509	0.309	0.283	0.282	0.282	0.282 / 0.027

set. For competitive performance, it may be worth applying these techniques on datasets *outside* of and *in addition to* those provide by WILDS. Second, the model architecture used here was an SVM model with input sentences encoded by Sentence BERT (Reimers & Gurevych, 2019). For truly comparable performance to the baseline models on WILDS, it would be ideal to fine-tune the `DistillBERT-base-uncased` model provided by HuggingFace (Sanh et al., 2020).

Future Work The empirical aspect of this work could easily be extended by repeating the experiments across a broader range of text classification datasets, adjusting the model (e.g., calibrating the confidence scores, smoothing the class weights, or fine-tuning a pre-trained BERT model), or extending the data augmentation strategies beyond the simple baselines used here. Particularly, diversity and traffic-aware sampling could be especially effective active learning algorithms for improving worst-group performance (Monarch, 2021; Sen & Yilmaz, 2020). On the self-learning front, there are multi-view training strategies that have shown to be more effective (Ruder & Plank, 2018). Stepping back from empirical evaluation, it would also be interesting to understand if there are properties of the dataset or model that can inform which data augmentation strategy is the appropriate choice. Mussmann & Liang (2018) found that for uncertainty sampling, there is a "strong inverse correlation between data efficiency and the error rate of the final classifier." This could explain why active learning was no more effective than random sampling on the Amazon dataset, for example, which may have more noisier labels than CivilComments. While this work serves as a useful starting point, it would be helpful for practitioners to have a more general theoretical framework for deciding which data augmentation approach to take.

Conclusions Overall, the objective of this work was to empirically compare active learning and self-learning methods for NLU in the context of domain adaptation. From experiments on the WILDS CivilComments and Amazon benchmarks for text classification, I found that active learning alone or a hybrid approach of self-learning and active learning lead to the best performance on overall test accuracy. However, the results are quite dataset dependent. On the Amazon dataset, even though active learning had the best performance, it did almost no better than data augmentation with random sampling. I also tested some non-conventional approaches, such as active learning with high-confidence examples, and found that this can be a beneficial strategy.

6 Acknowledgments

I would like to thank Tatsu Hashimoto for helpful discussions and comments on this work.

References

Borkan, D., Dixon, L., Sorensen, J., Thain, N., and Vasserman, L. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, 2019.

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., and Kurzweil, R. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018. URL `http://arxiv.org/abs/1803.11175`.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. Supervised learning of universal sentence representations from natural language inference data, 2018.

Du, J., Grave, E., Gunel, B., Chaudhary, V., Celebi, O., Auli, M., Stoyanov, V., and Conneau, A. Self-training improves pre-training for natural language understanding, 2020.

Duong, L., Afshar, H., Estival, D., Pink, G., Cohen, P., and Johnson, M. Active learning for deep semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 43–48, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2008. URL `https://aclanthology.org/P18-2008`.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks, 2016.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021.

Monarch, R. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Manning, 2021. ISBN 9781638351030. URL `https://books.google.com/books?id=bNo2EAAAQBAJ`.

Mussmann, S. and Liang, P. On the relationship between data efficiency and error for uncertainty sampling, 2018.

Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Peshterliev, S., Kearney, J., Jagannatha, A., Kiss, I., and Matsoukas, S. Active learning for new domains in natural language understanding, 2019.

Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pp. 61–74. MIT Press, 1999.

Ramponi, A. and Plank, B. Neural unsupervised domain adaptation in nlp—a survey. *ArXiv*, abs/2006.00632, 2020.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

Ruder, S. and Plank, B. Strong baselines for neural semi-supervised learning under domain shift. In *ACL*, 2018.

Saito, K., Ushiku, Y., and Harada, T. Asymmetric tri-training for unsupervised domain adaptation, 2017.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Sen, P. and Yilmaz, E. Uncertainty and traffic-aware active learning for semantic parsing. In *INTEXSEMPAR*, 2020.

Sethi, P., Savenkov, D., Arabshahi, F., Goetz, J., Tolliver, M., Scheffer, N., Kabul, I. K., Liu, Y., and Aly, A. Autonlu: Detecting, root-causing, and fixing nlu model errors. *ArXiv*, abs/2110.06384, 2021.