Deep Learning based vs Markov Chain based Text Generation for Cross Domain Adaptation for Sentiment Classification

Omar Abdelwahab and Adel S. Elmaghraby
University of Louisville, KY USA
{Omar.Abdelwahab, Adel.Elmaghraby@louisville.edu}

Abstract

Cross-domain adaptation for sentiment classification is the process of adapting a classifier that uses knowledge from one or multiple source domains and little data from the target domain to function with acceptable accuracy, precision, and recall on the target domain. One of the challenges facing cross-domain adaptation for sentiment classification is the limited availability of labeled samples in the target domain. In this paper, we introduce text generation in the target domain as a solution to provide a set of labeled data in the target domain then compare deep learning based text generators such as LSTM RNN and GRU RNN against Markov chain based text generators. We first use a rule-based classifier that utilizes knowledge from different source domains in labeling the unlabeled samples in the target domain (Kitchen Product reviews) then we have selected high confidence labeled samples for training LSTM RNN, GRU RNN and Markov chain based text generators. We have evaluated the deep learning based and Markov chain based text generators by measuring the fscores and accuracies of the end classifier when trained on the data generated from each of these models when tested on the kitchen benchmark test set.

Keywords: Sentiment Analysis, RNN, Cross-Domain Adaptation, Markov Chain, Text Generation

1. Introduction

Cross-domain adaptation for sentiment classification is an active research area that involves learning features from one or multiple source domains then applying the learned classifier on a different target domain. Most of the prior techniques depends on the idea of extracting domain independent features from source and target domains then using these extracted features as a bridge to transfer the domain dependent features from the source to be useful for the target domain's end classifier. As the domain independent features become better at predicting their co-occurring domain dependent features, the more powerful the adaptation process will become. We have not observed cross-domain adaptation approaches based on text generation, which led us to explore deep learning based feature extractors and Markov chain based text generators. Deep learning text generators need a very large data set, however, we wanted to see if the outcome of training deep learning feature extractors on a small data set of 30,000+ reviews. We will evaluate the deep learning based and Markov chain based text generators on the benchmark kitchen test-set of Blitzer et al. (2007) then we will discuss and conclude our results.

2. Literature Review

State of the art approaches such as Blitzer et al. (2007), and Pan et al. (2010) depend on extracting the relationships between ngrams in the source and target domains by modeling their correlations with pivot features. Pivot features occur frequently in both domains, and serve as good predictors of document classes. The extraction of pivot features was based on their mutual information with positive and negative source labels. Spectral Feature Alignment (SFA) outlined in Pan et al. (2010) on the other hand holds the same idea as SCL proposed in Blitzer et al, (2007). Bollegala et al. (2015) relies on selecting domain specific features (pivots) and domain independent features in each domain such that the domain independent pivot features have similar representations in both domains and that each pivot feature accurately predicts its co-occurring non pivot feature in its respective domain. Yu et al. (2016) proposed a document embedding technique on the sentence level that is similar to SCL in the way that it creates two additional tasks that help in learning a sentence level embedding that supposedly works across different domains. Wu et al. (2017) proposed an active learning based technique for cross domain adaptation for sentiment analysis where a general purpose sentiment lexicon is adapted to be useful in the target domain using little labeled target domain samples that were labeled in an active learning manner. The results presented in Wu et al. (2017) showed to surpass the results achieved by other techniques published in the past eight years when tested on the same benchmark test set. On the other hand, our proposed unsupervised approach was created to gain insight into the usefulness of the deep learning based text generation methods when having little unsupervised labeled samples (total of 8.4 MB in size in our case), for training the text generators/language models. Since the samples that are rule based labeled are not big in size for the RNN to learn generating complete sentences with context. In the following sections, we will highlight the components of our system then will discuss the results we have recorded when testing our system on the benchmark kitchen product reviews data set of the Blitzer et al. (2007) data set. We will compare between the best results achieved when using deep learning based text generation techniques with

Markov chain based text generation techniques. The test set that we tested on the Blitzer at al. (2007) balanced kitchen test set.

3. System Description

The system consisted of three parts as shown in Figure 1. The first part focused on the rule based sentiment labelling of the unlabeled kitchen product reviews of the Blitzer et al. (2007) data set. The high confidence positive and high confidence negative labeled samples were chosen and named seed samples. We will go through the labelling process in subsection 3.2. The second part involved training deep learning based recursive neural networks (LSTM and GRU), and Markov chain based text generators on the seed samples for generating positive and negative kitchen reviews. Finally, the third component focused on training a logistic regression model on the generated positive samples, negative samples and seed samples combined. We will go through these components in the following subsections.

3.1. Preprocessing

The unlabeled kitchen product reviews were preprocessed according to Abdelwahab et al. (2016). In addition to removing stop words and replacing positive bearing ngrams with a "positive" symbol and similarly replacing negative bearing ngrams with a "negative" symbol.

3.2. Unsupervised labeling of target samples

We have used a simple rule based technique in labeling the target domain samples then selected the high confidence samples from these labeled samples to be used for training the language models/text generators. The purpose of using a simple rule based labelling technique for labelling a small sample of the target domain reviews was to compare the tolerance of the deep learning based text generation techniques against that of the Markov chain based techniques when being trained on a data set that is not 100% accurate. We will evaluate the performance of each text generation technique based on the fscores and accuracies achieved by the end classifier when tested on the benchmark kitchen test set (Blitzer et al. 2007) after being trained on the data generated from each text generation technique separately. The rule based labelling algorithm is formed of the following steps. For each word in a review, a positive polarity score was calculated using the WordNet Library's (WordNet et al. (2010)) pos_score function.Unigrams that had no pos_score or neg_score were assigned 0. The positive polarity scores were summed up then divided by the number of unigrams that had pos_score to get the average positive polarity score for the whole review, which was then stored. For each word in a review, a negative polarity score was calculated using the WordNet Library's neg_score function. Unigrams that had no neg_score were assigned 0. The negative polarity scores were summed up then divided by the number of unigrams that had neg_score to get the average negative polarity score for the whole review which was then stored. If there were no negative polarity ngrams in the review, the average negative polarity score was set to zero. Likewise, if there was no positive polarity bearing ngrams in the review, the average positive polarity score was set to zero. The average negative polarity score was subtracted from the average positive polarity score to get the polarity score difference between the average positive and average negative scores. If the difference was greater than +0.1, the review was labeled positive and if the difference was less than -0.1, the review was labeled negative. Reviews that had a polarity score difference in between -0.1 and 0.1 were labeled as unknown and were not used in training the language models. The reviews that were given a positive or a negative label after the rule based labelling will be referred to as the "seed reviews" throughout the paper. The seed reviews were used in training the LSTM and GRU RNN models for text generation as will be shown in the next subsection. The following equations illustrate how the polarity score were calculated for each review. The term rev[i] in equations 1 and 2 stands for ith review in the data set.

$$AvgP = \frac{1}{p}\sum_{i=0}^{len(review)} pos\_score(rev[i]) \quad (1)$$

$$AvgN = \frac{1}{n}\sum_{i=0}^{len(review)} neg\_score(rev[i]) \quad (2)$$

$$Polarity_{Score} = AvgP - AvgN \quad (3)$$

3.3. Training GRU and LSTM Models

We have experimented with varying the hidden state vector size and the number of layers of a GRU RNN and LSTM RNN networks. We have experimented with hidden state vector sizes of 64, 128, 256 and 512. Afterwards we kept the hidden state vector size at 50 then varied the number of layers from 2 to 5 then 10. Which lead us to train one GRU RNN model and one LSTM RNN model per hidden state

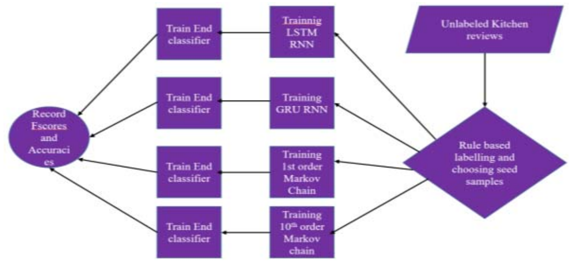
Train End classifier
Trainnig LSTM RNN
Unlabeled Kitchen reviews
Train End classifier
Training GRU RNN
Record Fscores and Accuraci es
Rule based labelling and choosing seed samples
Train End classifier
Training 1st order Markov Chain
Train End classifier
Training 10th order Markov chain

Figure 1: System Overview.

vector size per polarity (positive text or negative text). Then one GRU RNN model and one LSTM RNN model per number of layers per polarity. While we varied the number of layers and the hidden size, we have set the number of epochs to 2000, learning rate to 0.01, chunk length to 200 and batch size to 100. The GRU and LSTM implementation was based on Robertson et al. (2017)

implementation of the Character level text generation using GRU and LSTM.

3.4. Training Markov Chain Models

We have trained four Markov chain text generators. One Markov chain positive text generator of order 1, Markov chain negative text generator of order 1, Markov positive text generator of order 10 and Markov chain negative text generator of order 10. Each model was used in generating a balanced data set of 100,000 positive and negative reviews for training the end classifier. The purpose is to compare using text generated by Markov chain generators that were trained on poorly unsupervised labeled seed samples against deep learning based text generators by supplying the data generated by each technique to the same end classifier and comparing the accuracies and fscores achieved when using the markov chain based generated text vs the deep learning based generated text. The following equation represents the markov chain model with order m. Where n>m and in our case we have tried setting m to 1 then to 10 to train our order 1 and 10 models.

$$Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, .., X_1 = x_1) = Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, .., X_{n-m} = x_{n-m}) \quad (4)$$

3.5. Generating Text

After varying the number of RNN layers and feature vector size we have ended up with 14 GRU models and 14 LSTM models. Each model generated 25,000 product reviews which made the total number of GRU generated reviews and LSTM generated reviews to 350,000 each. The 350,000 reviews consisted of 175,000 positive reviews and 175,000 negative reviews.

3.6. Training the end classifier

After generating the positive and negative reviews, a logistic regression classifier was trained on the generated text combined with the seed samples and tested. Each generated training set used in training the model was parsed into a graphlab SFrame. Afterwards, each review was pre-processed as in section 2.1 then a TFIDF calculation is made for each review in the SFrame. The logistic regression model was trained on the TFIDF column which resulted in hundreds of thousands of unpacked features where we had to use L1 and L2 regularization to do dimensionality reduction and to avoid overfitting. We have performed grid search to find the near optimal L1 and L2 penalty values for the logistic regression model. The values taken by the L1 or L2 penalty variables during grid search were exponentially distributed. The graphlab library was used for training the logistic regression model. The following equations represent how the model was learned. Given a set of features xi, and a label yi∈{0,1}, logistic regression interprets the probability that the label is in one class as a logistic function of a linear combination of the features.

$$f_i(\theta) = P(y_i = 1|x) = \frac{1}{1 + e^{-\theta^T x}} \quad (5)$$

The objective function tries to minimizes the output of the sigmoid function in equation 4 while adding the two regularization terms l1 and l2 adding the two regularization terms l1 and l2 penalties using mini batch gradient descent. Where θ is the weight matrix.

$$min_\theta \sum_{i=1}^{n} f_i(\theta) + \lambda_1 ||\theta||_1 + \lambda_2 ||\theta||_2 \quad (6)$$

3.7. Markov chain training on small sample of labeled target domain data

We have trained additional Markov Chain text generators on 10% (160 positive reviews and 160 negative reviews) of the target domain labeled kitchen data set in Blitzer et al. (2007) for the purpose of highlighting the major improvement in the performance of the Markov Chain text generators when training it on a small sample of 100% correctly labeled data samples. We have trained two markov chain (one model with order 1 and the other with order 10) positive text generators and similarly two markov chain based negative text generators. Each generator produced 50000 reviews. We have combined the positive and negative reviews generated by the order 1 generators into one data set and named it Markov_Labeled_1 and likewise we have combined the positive and negative reviews generated by the order 10 models into one data set called Markov_Labeled_10. We have added the 10% labeled reviews to each of these two data sets and supplied the end data sets to the classifier in section 2.6. Tables 2 and 3 show the results that we got and it shows clearly that there was a double digit improvement in accuracy and fscore when using text generated from Markov chain text generators trained on 100% correctly labeled samples than using text generated from Markov chain text generators that were trained on a larger data set that was labeled by a simple unsupervised rule based labelling technique that has an accuracy of 71% on the Blitzer Kitchen test set. The size of the seed samples that were used in training the Markov Chain text generators in section 2.4 were 33567, which consisted of 8093 negative labeled reviews and 25474 positive labeled reviews.

4. Experiments and Results

In this section, we will get to showcase the results we have recorded while varying the text generator's architecture (GRU vs LSTM), Hidden state vector size (64, 128, 256, 512) and number of hidden layers (2, 5, 10). The evaluation criteria for the text generation methods will be task oriented. Which means that the quality of the generated text will be evaluated based on the best accuracy achieved by the end classifier after grid search on the balanced kitchen test set. We will show the best accuracy achieved with each generated training set, L1 and L2 penalty combinations on the balanced Kitchen test set of the Blitzer et al. (2007). All Fscores and accuracy values were averaged over 10 trials.

4.1. Results – Deep Learning vs Markov Chain

Tables 1 and 2 show that the deep learning based text generators resulted in better performance when trained on the noisy/unsupervised labeled seed samples when compared against the Markov chain text generators. However, Table 3 show that when training the Markov chain text generators on 100% correctly labeled samples, it performs way better than when it is trained on weakly labeled samples which means that deep learning based text generators proved to be more resilient to poorly labeled training samples.

Training Data	L1	L2	Acc	F-Score
GRU_Full	10	100000	0.74	0.75
LSTM_Full	100	100000	0.77	0.74

Table 1: End Classifier Accuracy when using LSTM_Full and GRU_full data sets.

Training Data	L1	L2	Accuracy
Markov_order1	100	100	0.54
Markov_order10	10	1000	0.502

Table 2: End Classifier Accuracy when using Markov_order1 and Markov_order10 (seed revs)

Training Data	L1	L2	Accuracy
Markov_order1	1000	100000	0.71
Markov_order10	10	10	0.72

Table 3: Markov Chain Text Generators (10% Target labeled).

5. Conclusion

The results that we have obtained suggest that Markov Chain based text generators have little tolerance to incorrectly labeled reviews in the training set. As the accuracy of our unsupervised rule based classifier is around 71% on a balanced test set. Therefore, a percentage of the labeled seed reviews contain false positives or false negatives which lead to damping the performance of the Markov chain text generators as we have shown in the results section. On the other hand, deep learning based models have higher resilience towards the presence of false positives or false negatives in the training set which showed that the accuracies and fscores achieved by the end classifier did not dip to the fifties as in the case of using the Markov chain text generators. We will explore using text generation on a larger training set in the future while exploring other Neural networks architectures like the GAN networks. We will explore using text generators that make use of word level and character level features as Xie et al. (2017) have discovered that language models that make use of character level features and word level features perform better than models that depend on character level features only or word level features only.

6. References

[1] Fangzhao Wu, Yongfeng Huang, and Jun Yan, *Active Sentiment Domain Adaptation*, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1701–1711 Vancouver, Canada, July 30 - August 4, 2017.

[2] Jianfei Yu, and Jing Jiang, *Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification*, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, November 1-5, 2016.

[3] John Blitzer, Mark Dredze, and Fernando Pereira. 2007. *Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification.* Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.

[4] Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang and Zheng Chen. 2010. *Cross-domain sentiment classification via spectral feature alignment.* Proceedings of WWW '10 Proceedings of the 19th international conference on World wide web Pages 751-760.

[5] Princeton University "About Word-Net." WordNet. Princeton University. 2010. [online] Available at: https://wordnet.princeton.edu/citing-wordnet

[6] Natalia Ponomareva and Mike Thelwall. 2012. *Do neighbours help?: an exploration of graph-based algorithms for cross-domain sentiment classification.* EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning Pages 655-665.

[7] Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015 *Unsupervised Cross-Domain Word Representation Learning.* Proceedings of e 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 730–740.

[8] Omar Abdelwahab and Adel Elmaghraby. 2016. *UofL at SemEval-2016 Task 4: Multi Domain word2vec for Twitter Sentiment Classification.* Proceedings of SemEval-2016, pages 169–175

[9] Xiang Zhang, and Yann LeCun. 2016. *Text Understanding from Scratch.* arXiv:1502.01710

[10] Sean Robertson GitHub. (2017). *spro/char-rnn.pytorch*. [online] Available at: https://github.com/spro/char-rnn.pytorch

[11] Stanley Xie, Ruchir Rastogi, and Max Chang. 2017. *Deep Poetry: Word-Level and Character-Level Language Models for Shakespearean Sonnet Generation.* [online] Available at: https://web.stanford.edu/class/cs224n/reports/2762063.pdf