Cross-Domain Product Classification with Deep Learning

Luke de Oliveira,[1] Alfredo Láinez Rodrigo,[1] and Akua Abu[1]

[1]*Stanford University ICME, CS229 Final Project*

In our final project, we took on the challenge of cross-domain classification–the adaptation of a model suitable for one domain to one or more different domains with identical features. Specifically, we developed a cross-domain classification schema to predict product categories in tweets and eBay reviews based on user reviews from Amazon. We first developed a set of classical machine learning classifiers that performed reasonably well on our principal data set of Amazon products and then applied the same classifiers to predict product categories from Twitter tweets and eBay reviews. We then developed Deep Learning models to approach to our cross-domain classification problem and compared performance. In what follows, we walk step by step through the procedure, illustrate our results, then discuss future work. We show that while Deep Learning definitively helps models work out-of-domain, considerable work needs to be done to create a context-free model of language that can work across domains.

I. INTRODUCTION

There is no longer a boundary between social media, advertising, and consumer culture. The intersection of these different spaces has resulted in an immesurable increase in the amount and prevalence of user data. Users develop online identities that span across many platforms, from Facebook to Twitter to YouTube and to consumer communities such as eBay and Amazon. An open problem of great relevance to these overlapping social spheres is that of *cross-domain learning* – that is, given an estimated model suitable for one domain, can the model perform in an entirely different context given identical features? Using a dataset of Amazon products labeled within the Amazon hierarchical categories, we have developed a set of classifiers that predict product categories based on users' reviews. Then, we apply the same predictors of user content to different domains to see how these models behaves in a different context over different underlying distributions of features. The key challenge is that we impose no prior over the target distribution – we simply aim to develop a model that is not entirely dependent on a training distribution. Specifically, we have predicted product categorization using *tweets* from Twitter and reviews from eBay, and we determine the validity of an out-of-context classifier using carefully chosen metrics. Finally, we have engineered two Deep Learning models, and use them to illustrate a potential new application for Deep Architectures – we outperform classical approaches in this task, showing adaptability to new domains.

II. SETUP

A. Data

This project makes use of a principal data set of Amazon product reviews and an Amazon product hierarchy. Secondary data sets include product-related Twitter data (tweets), and ebay product reviews – each of which are hand labeled. The main Amazon data sets consists of descriptions and 10 reviews for each of the top 10 products in every Amazon product category, comprising a hierarchy of roughly 400,000 products ranging across 40,000 categories. The secondary data set is of eBay and Twitter data: the set contains tweets and reviews along with an annotated Amazon category. For our classification purposes, we used the 24 main categories in the Amazon hierarchy, merging the main categories *KindleStore* and *Books*. Then, our classification operates over text (reviews or tweets) labeled by a main Amazon category.

The data used was saved in JSON format, including node categorization in the Amazon product hierarchy for each product. We built a parser and graph creation codebase that allowed us to extract this hierarchy in the form of a tree and traverse it for each product, allowing us to extract labels at any depth. Specifically, we utilized the nodes with $d = 1$ away from the root node – i.e., the main categories. For the purposes of our investigation into cross domain text classification, we care less about a fine study of large-scale hierarchical clasification and more about whether or not we can find methods that are valid across the domain – in some sense, a problem completely independent from the former.

B. Evaluation

We are working in a classification space with 24 different groups with *highly* imbalanced classes. In order to compensate for this, we utilize F_1-scores averaged out across classes, where

$$F_1^{(j)} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

is the F_1 score for class j. We compute precision and recall for each class in a *one-vs-rest* approach. Our overall F_1 score (for J classes) is then

$$F_1 = \frac{1}{J} \sum_{j=1}^{J} F_1^{(j)}.$$

More generally, one can imagine a scenario in which we care about some convex combination denoted by a vector

c, where

$$F_1^c = \sum_{j=1}^{J} c_j F_1^{(j)},$$

and each c_j denotes our "relative cost" of an F_1 error on class j. However, we opt for a standard mean to give each class equal influence in terms of how well we predict its membership. Traditional classification error was not used as this provides us with a better metric given our imbalances.

It is important to note that a few classes are heavily underrepresented in the data, even considering the whole dataset. Hence, the average computed as a quality measure will be affected by these classes, and will be significantly skewed downward, moving the bias away from well-represented classes.

C. Features

As pure text was the raw input to our model, we utilized a bag-of-words representation of our review texts, incorporating *tf-ifd* and removing common stop words to improve performance and direct our attention to semantically relevant aspects of cross domain language generalization. The features thus corresponded to a sparse matrix of word frequencies from our selected corpus. The inverse document frequency (ifd) improved the results ruling out words commonly present in the corpus. Moreover, we incorporated an n-gram approach into our model to better take account of differences in meaning resulting from contiguous sequences of words. For $n > 1$, run-times were longer, but we obtained a slight increase in test set performance with worse results for cross-domain classification. This is consistent with *a priori* intuition regarding these features, as n-grams with $n > 1$ provide a very refined look at language structure, meaning any model learned will not generalize well.

III. CLASSICAL METHODS

As a first classification step, we selected three well-known algorithms for the task of text classification: Naive Bayes, Stochastic Gradient Descent and Support Vector Machines, followed by a grid search / CV for parameter optimization. In particular, we utilized Naive Bayes with Laplacian smoothing and SGD with a *modified huber* loss function and $\alpha = 0.001$. Also, the SVM is a linear SVM which follows a *one-vs-the-rest* multiclass strategy, since other SVMs with *one-vs-one* strategies are too time consuming for our space of 24 categories. In particular, given the constraints of `scikit-learn`, utilization of any Kernel Machine requires the estimation of $\binom{J}{2} = 276$ (in our case) Kernel Machines – a prohibitive cost.

During the implementation and evaluation, we noticed that these classically derived "shallow" methods

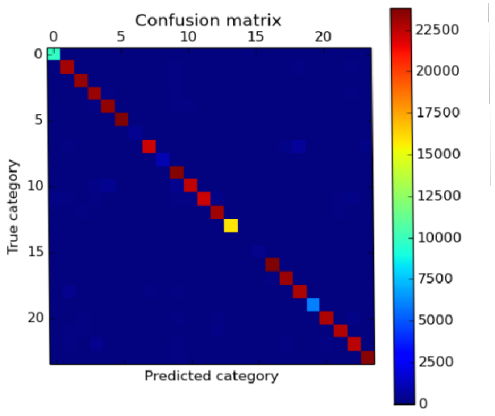

FIG. 1: Confusion matrix for predicting categories in the Amazon dataset using a SVM with 2-grams bag of words

presented certain bias towards the majority class. Since Amazon started as a bookstore, the number of products categorized as books is about 30% of the entire dataset, with other categories also comprising a large proportion of the training examples. At the opposite end of the spectrum, we have a pair of product classes comprising less than 100 products in total. The problem of training with such overrepresented classes made a Naive Bayes classifier simply predict the class "Books" for every given example.

In order to avoid this training bias, and considering the large amount of data available, we loaded the whole dataset for training and testing purposes and selectively pruned it in order to balance the classes for training. While this does not solve the underrepresentation of some classes, it provides more balance to the majority of them and utilizes all the examples available of the less common ones. In FIG 1 we can see a confusion matrix illustrating the distribution of the classes after pruning and classification for the Amazon dataset.

IV. DEEP LEARNING

As our foray into classical methods proved, a simple application of a classifier is not appropriate for estimating a model with the expectation of out-of-domain performance. To examine how we can construct a model that can learn features that can prove useful both inside and outside the training domain, we look to deep learning. In particular, we look to ways in which we can provide *regularization*, as this will help our models adapt to different domains. The challenge in our chosen problem is that we have no prior on the language distribution in the target domain. A series of structures / paradigms were decided upon that would be able to test the relevance of Deep Learning for this particular cross-domain classification problem. We consider stacked autoencoders of regular and denoising variety, and a "stacked thesaurus," as we call it – a novel modification to Dense Cohorts of Terms. These are all trained in an unsupervised fashion, with supervised fine-tuning

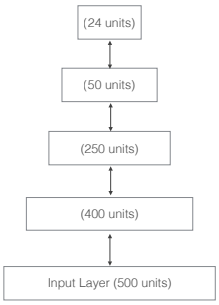

FIG. 2: Network structure used for Autoencoders

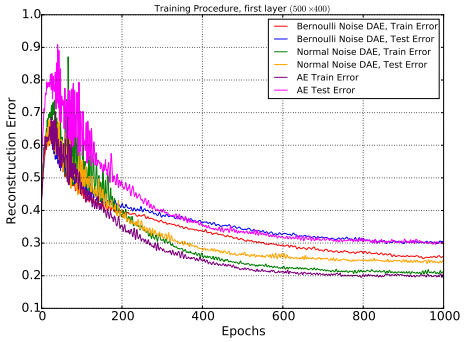

FIG. 3: Error Curves for training – first layer of deep model.

applied later. Since Deep Models are expensive to estimate, we utilize a subset of possible features from *tf-idf* matrices, restricting ourselves to the top 500 features for Autoencoder-based methods.

For all deep learning methods, we use Stochastic Gradient Descent as our training algorithm. In addition, we use momentum to speed up training. All learning rates, momentum parameters, and noising / regularization parameters are found using grid search. Using our own implementation, we rely on `scipy/numpy` to handle matrix math.

A. Stacked Autoencoders

An autoencoder is a tranformed linear mapping that creates a "bottleneck" of dimensionality. Formally, consider a data vector $x \in \mathbb{R}^D$. Suppose we want to find a representation $r \in \mathbb{R}^d$, $d < D$. An autoencoder consists of an encoder, which is a pair (W_1, b_1), with $W_1 \in \mathbb{R}^{d\times D}, b_1 \in \mathbb{R}^d$, and a decoder, which is a pair (W_2, b_2), with $W_2 \in \mathbb{R}^{D\times d}, b_2 \in \mathbb{R}^D$. We also need two mappings $f, g : \mathbb{R} \longrightarrow \mathbb{R}$, that can be applied elementwise to vectors and matricies. The encoder then generates a new space by the map $\varphi(x) = f(W_1 x + b_1)$, and the decoder reproduces the original vector by the map $\rho(x) = g(W_2\varphi(x) + b_2)$. To ensure this new space represents the original data vector well, we would like to find W_i, b_i such that $x \approx \rho_\theta(x)$.

B. Stacked Denoising Autoencoders

The basic autoencoder aims to create a lower dimensional coding that contains all information about the original data vector. Specifically, over a data batch $X \in \mathbb{R}^{D\times n}$, a standard autoencoder defined by the loss function

$$L(X;\theta) = \frac{1}{2n}\|X - g(W_2 f(W_1 X + b_1 \mathbf{1}^T) + b_2 \mathbf{1}^T)\|_F^2,$$

has no regularization terms, and aims to build the *best reconstruction* possible. However, we have no guarantees as to whether or not the locally optimal mapping we find to generate this coding is robust – that is, how sharply does the reconstruction error varies around a given data vector x. In order to force the autoencoders we stack to construct *robust* representations, we require that the autoencoders be able to reconstruct a clean version of x from a corrupted version $x + \delta x$. Specifically, this is a *denoising* requirement, and acts as a regularizer to local changes in distributions.

The denoising autoencoder is defined by the loss function

$$L(X;\theta) = \frac{1}{2n}\|X - g(W_2 f(W_1 \tilde{X} + b_1 \mathbf{1}^T) + b_2 \mathbf{1}^T)\|_F^2,$$

where $\tilde{X} \sim P(\tilde{X}|X)$ is a stochastic, corrupting mapping. Common examples of corruption mappings are

- $P(\tilde{X}|X) = X \odot B, B_{ij} \sim$ Bernoulli(p) (i.e., a binary mask)
- $P(\tilde{X}|X) = X + N_{(0,\varepsilon^2)}$, where $[N_{(0,\varepsilon^2)}]_{ij} \sim N(0, \varepsilon^2)$

We greedily train each denoising autoencoder, *stacking* as we go along. At the top level of such a stack, we hope to have a set of features that is both robust in construction and captures correlations between words that are useful across domains.

Now, we examine the training procedure of the first layer, using different variants of Autoencoders with different regularization procedures to understand what each layer will accomplish. Consider FIG. 3. We see that we have better test sample performance with denoising autoencoders with gaussian noise when compared with the normal-type autoencoders. This makes sense – our generalization to examples should be better with imposed denoising criteria. We proceed with a similar training procedure layer-by-layer – helping the network learn a stacked, hierarchical representation of our training sample.

C. Stacked Thesauri

We present a modification of Dense Cohorts of Terms, as introduced in [5]. In particular, we aim to reconstruct smaller and smaller subsets of our vocabulary in each

layer, corresponding to higher and higher levels of semantic meaning. Instead of creating a coding that contains all information contained in data vectors, we instead want to create a mapping that finds ways to map corrupted, rare words into a set of more common words, in a sense acting as a *thesaurus*. Let our vocabulary be of size D, and assume we obtained a Bag-of-Words representation $X \in \mathbb{R}^{n\times D}$, there all nonzero entries have value 1 for ease. Now, suppose we want a Deep Model to learn what it means for words to be synonyms – that is, can we link many uncommon words such as "exquisite," "marvelous," and "impeccable" to a more common word like "good"?

We take the top $R < D$ words in terms of frequency, and construct a (binary) Bag-of-Words representation $P_{(R)} \in \mathbb{R}^{n\times R}$. Now, we ask – can we find a mapping such that $\sigma(XW_1) \approx P_{(R)}$, where $\sigma(\cdot)$ is the standard sigmoid? In order to be resistant to any differences in distribution, we can corrupt X with Bernoulli noise – that is, we set each non-bias feature to zero with some probability p.

To stack, we use the output of the previous layer, call it $Z \in (0,1)^{m\times R}$, as input, and we simply now choose a smaller target "Thesaurus" size – let this be R'. We now wish to find a mapping $\sigma(ZW_2) \approx P_{(R')}$. We can repeat this in a greedy manner until we have the desired structure We utilize the same structure as outlined in FIG. 2.

V. RESULTS

For the following results, the classifiers were trained on a total set of 400,000 training examples and 100,000 test examples. The data was pruned in a preprocessing step from an even bigger dataset in order to reduce class imbalance, as explained before. The cross-domain sets contain 62,000 *tweets* and 33,000 eBay reviews.

Method	Train	Test	eBay	Twitter
Naive Bayes	0.54	0.51	0.31	0.18
SGD	0.54	0.53	0.30	0.15
SVM	0.77	0.67	0.29	0.16

TABLE I: F_1-scores for classical methods

In TABLE I, we can see how the best Amazon training and Amazon test results are yielded by the most sophisticated algorithm, the Linear support vector machine. However, we take note how this improvement do not translate cross-domain, where the performance even decreases. This is a clear signal of the fact that there are distinct characteristics in different text domains. In FIG. 4 we can appreciate how the F1 classification scores for the domains in eBay and twitter stop increasing rather soon, even worsening when the models are fitting better the training domain.

Next, consider results (TABLE II) from Deep Learning using the aforementioned pretraining methods – Stacked Autoencoders (SAE), Stacked Denoising Autoencoders (SdAE), and Stacked Thesauri (SThi).

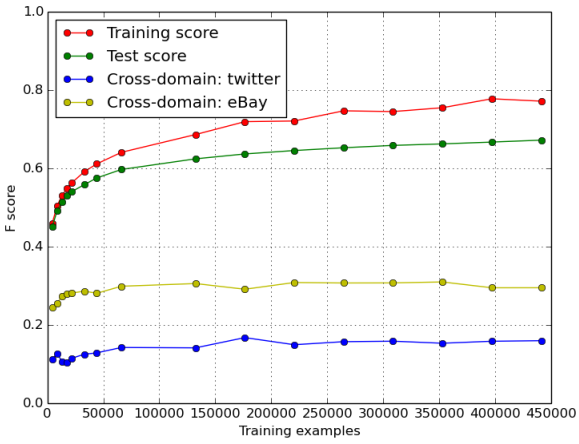

FIG. 4: Learning curve for the linear SVM

	Method	Train	Test	eBay	Twitter
Simple Softmax	SAE	0.85	0.801	0.31	0.198
	SdAE	0.85	0.83	0.371	0.221
	SThi	0.84	0.81	0.341	0.202
Fine-tuning	SAE	0.871	0.82	0.29	0.15
	SdAE	0.862	0.831	0.36	0.21
	SThi	0.849	0.827	0.381	0.229

TABLE II: F_1-scores for Deep Learning

VI. DISCUSSION

It is clear from our results that it is not possible to fully extrapolate a model to a different domain and expect similar behavior. While the combination of our techniques did allow for accurate cross-domain classification for a few subsets of product reviews, we were not able to develop a completely accurate model that could be extrapolated across all domains. Among classical classifiers used, support vector machine produced the strongest results on the original data set; however, as with the other classifiers, cross-domain results were relatively weak.

Particularly we experienced particularly low performance for the Twitter data set, which can be partially attributed to the inability of our model to fully take into account the slang-based vocabulary. While Amazon reviews tend to be reasoned opinions about a product, *tweets* are extremely short texts using a very particular vocabulary, many times utilizing hashtags and varied abbreviations – semantic units that were in no way present over the data the model was trained over. More surprising is the difference in the performance of the classifiers in the realm of eBay product reviews, where one would suppose a strong resemblance to those of Amazon. However, from the results, and assuming that the data

is correctly labeled, we can conclude that the users vocabulary when reviewing products in these domains are different.

It is important to note that Deep Learning did non-negligibly improve the performance of out classifiers on the original data set while also increasing results on the cross-domain data sets – most notably, the eBay data set – when compared to "shallow" counterparts.

Our results from deep learning fit nicely into our beliefs with respect to how we think regularization will help out model extrapolate to a priorless, unseen domain. In particular, note that pure SAE with fine tuning performed best on the training set of Amazon – an unsurprising fact, seeing as though SAE try to fit an exact reconstruction during training time. Note that with this excellent training set performance comes serious difficulties generalizing to our cross-domain targets – in particular, we notice in TABLE II that we obtain F_1 scores that are just as bad as the classical Linear SVM.

However, we make note of what happens when we impose the restriction that our model must denoise as well as reconstruct. In particular, we notice that though our Amazon training set accuracy decreases, our performance on the Amazon testing set increases. In this regime, it is impossible to overtrain the pretrained model – every training example is noised in a different manner each time. Also noteworthy is the increase in cross-domain performance, though we obtain better generalizability by simply appending and training a softmax layer.

Most interesting, we see that our Stacked Thesauri yield the best cross-domain performance. We postulate that this is due to the fact that the representations learned by SThi are dense in commonly used terms, which (we believe) are more constant across the domains we considered than the language distribution at large. The ability to encode synonyms allows greater flexibility when dealing with limited vocabularies, and reduces reliance on product-specific jargon or vernacular to make predictions.

VII. FUTURE WORK

Firstly, with greater computing resources, it would be interesting to improve the efficiency of our schema and apply it to the complete set of labels from the Amazon product hierarchy. Moreover, our project could be extended in a follow-up study to detect differences in the vocabularies used in the different domains, and fine-tune our approaches to overcome these differences and increase cross-domain classification performance. In addition, it would be interesting to see if we could design some sort of generative model related to Conditional Restricted Boltzmann Machines which would help us condition on small subsets of the distribution as we learn more about the different domains we apply to.

We could also modify our principal data set, perhaps training on the domains with more specific vernacular such as Twitter with its slang-based vocabulary. In addition, we would be interested in using models such as the over-replicated softmax model as an alternative approach to feature engineering.

VIII. CONCLUSION

We examined the cross-domain classification of online user behavior in terms of text. In particular, we asked: can we use labeled Amazon data to extrapolate out-of-domain and determine what products individuals in Twitter and eBay are discussing? While classical methods performed relatively well on the original data set, they were inherently unable to translate this performance to the cross-domain problem. Our deep models with fine-tuning provided a considerably higher degree of accuracy on the original data set and on the cross-domain problem. In particular, we showed promising results using Stacked Thesauri – a method that relies on stacking layers combining semantically similar words. However, we have found that without prior knowledge of the language distribution, it is a difficult task to extrapolate machine learning models to different domains, even with the powerful and generalizable toolbox of deep learning.

IX. ACKNOWLEDGEMENTS

We wish to acknowledge Professor Ashutosh Saxena and Aditya Jami for the very interesting project idea and for providing the data used.

[1] Sha Fei Gong Boqing and Kristen Grauman. Overcoming Dataset Bias: An unsupervised domain adaptation approach. *NIPS*, 2012.

[2] Yun. Jiang and Ashutosh Saxena. Discovering Different Types of Topics: Factored topic models. *Cornell*, August 2012.

[3] Nitish Srivastava, Ruslan Salakhutdinov, and Geoffrey E. Hinton. Modeling documents with deep boltzmann machines. *CoRR*, abs/1309.6865, 2013. URL `http://arxiv.org/abs/1309.6865`.

[4] Beal M. Teh Y., Jordan M. and D. Blei. Hierarchical Dirichlet Processes. *Berkeley*, March 2005.

[5] Zhixiang (Eddie) Xu, Minmin Chen, Kilian Q. Weinberger, and Fei Sha. From sbow to dcot marginalized encoders for text representation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1879–1884, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1156-4. doi: 10.1145/2396761.2398536. URL `http://doi.acm.org/10.1145/2396761.2398536`.