Cross-Domain Sentiment Classification using Vector Embedded Domain Representations

Nicolaj Filrup Rasmussen
IT University of Copenhagen
nicr@itu.dk

Kristian Nørgaard Jensen
IT University of Copenhagen
krnj@itu.dk

Marco Placenti
IT University of Copenhagen
mapl@itu.dk

Thai Wang
IT University of Copenhagen
twan@itu.dk

Abstract

Due to the differences between reviews in different product categories, creating a general model for cross-domain sentiment classification can be a difficult task. This paper proposes an architecture that incorporates domain knowledge into a neural sentiment classification model. In addition to providing a cross-domain model, this also provides a quantifiable representation of the domains as numeric vectors. We show that it is possible to cluster the domain vectors and provide qualitative insights into the inter-domain relations. We also a) present a new data set for sentiment classification that includes a domain parameter and preprocessed data points, and b) perform an ablation study in order to determine whether some word groups impact performance.

1 Introduction

In recent years the amount of text data has been growing exponentially. With the growth of social media and the success of e-commerce, large websites such as Amazon have developed great interest in online user reviews, which the users are able to write about every product. The task of classifying review sentiments poses little challenge for humans. However, with large amount of data comes the necessity of automating such tedious classification tasks, which proves to be a challenge that needs careful development and consideration.

This challenge arises because reviewers use different registers when writing reviews across product domains. Consequently, machines are not well equipped to catch those differences. The differences are easy to detect when users reference domain-specific words or other products in the same category. This belongs to a domain-specific knowledge that a neural network needs to learn in order to fully understand the differences and the similarities within the context of the reviews. Thus, for a classifier to perform well at this task, it may be important to include a way for it to represent these domains.

In this paper, we propose a novel, yet effective way of representing domains in sentiment classification. Using this representation we will show that it is possible to visualize relations between domains. This will allow clustering of similar product categories in the feature space of the domain representations. We also attempt to evaluate whether these groupings are similar to those made by humans.

Implementing the domain representations as a vector embedding provides a simple and elegant solution compared to previous solutions like Peng et al. (2018) and Liu et al. (2018). The aim of this paper is to de-

Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing, pages 48–57

velop a cross-domain sentiment classification model achieving comparable performance to those of existing methods, with the added benefit of insights into the inter-domain relations. Additionally we have developed a small but interesting data set building on top of McAuley et al. (2015). Our main contribution being balancing and preprocessing the data, as well as adding the domain in an easy to parse way. The data set is available at https://static.nfz.dk/data.zip.

2 Related Work

Sentiment classification is a well-established and -researched field within natural language processing. Pang et al. (2002) tested three machine learning algorithms for performing sentiment classification on sampled movie review data. Go et al. (2009) also tested three machine learning algorithms. However, they wanted to test the performance on documents that include emoticons which are sampled from Twitter data.

Including domain knowledge in models is nothing new either. Alumäe (2013) has used domain information to train a language model that responds differently in different domains. Tilk and Alumäe (2014) presented a Recurrent Neural Language model used for training an Automatic Speech Recognition system. Ammar et al. (2016) presented a multilingual model for parsing sentences, in which languages are encoded in their own variable.

Peng et al. (2018) presented a method for performing cross-domain sentiment classification on sparse labeled domains using domain adaption. The paper proposes a co-learned model which uses target specific features and domain invariant features to classify sentiment. Performing multi-domain classification has received little attention within Natural Language Processing. Nam and Han (2016) used multi-domain classification for visual tracking. More recently Jia et al. (2019) performed cross-domain NER by using a parameter generation network to learn domain and task correlations. They did this in an unsupervised setting and achieved state-of-the-art results among the supervised counterparts.

Li and Zong (2008) proposed two different ways of performing multi-domain sentiment classification. One was combining the domains at the feature level and the other was combining it at the classifier level. When combining at the classifier level, one could also think of using a multiple classifier system (MCS). Li et al. (2011) proposed using an MCS for performing multi-domain sentiment classification. Here they applied both fixed and trained rules for combining the output of the classifiers. Liu et al. (2018) proposed that multi-domain sentiment classification can be done for both abundant and sparse labeled domains, using domain specific representations. They produced domain general representations using a Bi-LSTM and learning domain descriptors using an attention network. Moreover, they trained a separate domain classifier using the domain general representations. The whole model was trained using a minimax game.

The model proposed in this paper uses a structure similar to the latter but distinguishes itself by being simpler, allowing for faster training or training on more modest hardware. The proposed architecture excludes the domain classifier and approaches the task of learning domain specific features using vector embeddings. In addition, great emphasis is put on the embedded vectors produced during training and what they can tell us about the relations of the domains.

3 Methodology

The proposed model is composed of three blocks, as shown in Figure 1.

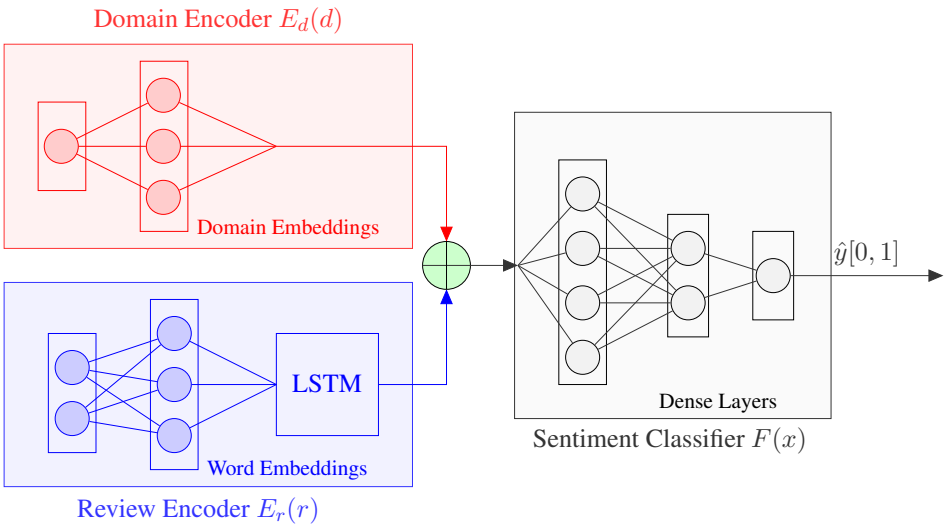


Figure 1: Our proposed model architecture

- Domain encoder $E_d(d)$
- Review encoder $E_r(r)$
- Sentiment classifier $F(x)$

Two of the blocks, $E_d(d)$ and $E_r(r)$, work as encoders, encoding the incoming data and making it usable for the final classifier. The classifier is a simple feed forward neural network that uses the encoded data to classify the review as either positive or negative. The blocks are described in more details in Sections 3.2 to 3.4.

3.1 Data set generation

The data used in this paper has been collected from Amazon (McAuley et al., 2015). All of the collected reviews are in English.

12 domains were selected and 200,000 reviews were sampled from each (100,000 positives and 100,000 negatives). The reviews were sampled at random. In total, 2.4 million reviews were collected from the 12 domains (See Table 1). The data set is structured such that each data point consists of the following four attributes: y, $review$, $topic$, $length$. Here y corresponds to the correct sentiment label, $review$ is the tokenized review which is also segmented into sentences, $topic$ is the domain label and $length$ denotes the number of sentences in the review.

Category	Negative	Positive	Total
Automotive	100K	100K	200K
Baby	100K	100K	200K
Beauty	100K	100K	200K
CD's and Vinyl	100K	100K	200K
Cell Phones	100K	100K	200K
Electronics	100K	100K	200K
Personal Care	100K	100K	200K
Movies and TV	100K	100K	200K
Office Products	100K	100K	200K
Sports	100K	100K	200K
Toys and Games	100K	100K	200K
Video Games	100K	100K	200K

Table 1: List of domains sampled from (McAuley et al., 2015)

First and foremost, we concatenate each review text onto its summary text, effectively treating the summary text as the first sentence. This is done to use all of the data available.

When the review and summary have been concatenated, we segment the text into sentences using a sentence segmenter from NLTK.[1] The segmenter is trained on the full data set in order to fit the style and abbreviations used in the reviews. This enables identification of a part-of-speech (POS) tag for each word using a pre-trained POS tagger. The POS tagger is from NLTK and uses the Penn Treebank tagset. Using these tags it is possible to create new data sets with certain word groups ablated. Each sentence is subsequently tokenized into words with the exception of digits and some punctuation.[2] The sentence split is not used in this model but is available in the data set for future research.

Data set	Unique tokens
Full data set	27,112
Ablated adjectives	24,225
Ablated stop words	26,962

Table 2: Vocab sizes of the constructed data sets

Lastly, we choose to only include words that occur more than 100 times in the vocabulary and represent the discarded tokens with an <UNK> token. This is done purely to limit the size of the vocabulary. The vocabulary size for each of the three data sets can be seen in Table 2. The three vocabularies mentioned here are the full set with all word groups: the set where adjectives have been removed and the set where stop words have been removed. The stop word list used is from NLTK.

3.2 Domain Encoder

The Domain Encoder takes one of the 12 domains presented in Table 1 and encodes it into its own internal representation. The idea behind this architecture is to make the model learn its own representations for each domain (Mikolov et al., 2013) and how they relate to each other. There are several ways of implementing this representation, but the choice to use vector embeddings was made because it is simple and because they are easy to interpret compared to some of the more abstract methods of representation. Each domain is represented by a 50-dimensional vector that is learned during training. A 50-dimensional vector is chosen to match the size of the pre-trained word vectors in the Review Encoder (Section 3.3). The domain representations are randomly initialized using a uniform distribution.

3.3 Review Encoder

This encoder takes a vector of variable length for the input layer, meaning that the layer adapts itself continuously according to the sentence given as input. We set a threshold to 500 tokens, effectively truncating sequences longer than this. Each token in the sequence is represented by pre-trained GloVe embeddings (Pennington et al., 2014). The output of the Review Encoder is an LSTM layer with 64 neurons (Hochreiter and Schmidhuber, 1997).

3.4 Sentiment Classifier

The two previous outputs (from the Domain Encoder and the Review Encoder) are concatenated together, resulting in a vector of length 114 as input for the Sentiment Classifier. The Sentiment Classifier is a standard feed forward neural network, where the first hidden layer consists of 128 neurons, which is followed by a second hidden layer consisting of 64 neurons. Finally, we have a single neuron which outputs the probability of the sentence being positive. The probability is rounded to the nearest integer, where 1 is considered positive and 0 is negative.

[1] http://www.nltk.org

[2] Included punctuation: **...** **:** **;** **,** **(** **)** **!** **?** **”** **’**

3.5 Hyper Parameters

The model's hyper parameters are tuned using random search. The implementation is provided by an open source project (Autonomio, 2019).

The data set used for the hyper parameter tuning consists of 150,000 samples, which were sampled from the full training set. From this, 50,000 samples are sampled for validation and the remaining 100,000 samples are used for training. This split is performed 5 times resulting in 5 different validation/train splits. When sampling we make sure the resulting data sets are balanced between the sentiment labels. For each run, one of the splits is selected at random in order not to fit to a specific data set. We evaluate each set of parameters based on the accuracy score achieved by the model on the sampled validation set.

3.6 Experiments

We analyze the relevance that the domain embedding has in our model by testing our model twice, once with the Domain Encoder and once without. These two experiments are run 4 times and the mean is reported. Furthermore, we test the proposed architecture by ablating adjectives and stop words separately. In total we conduct 4 different experiments on the model. We split the data set such that we have around 2 million reviews for training, 200,000 for validation and 200,000 as a hidden test set. In all four cases we run 10 epochs of training while validating in between epochs. Finally, we evaluate on the hidden test set.

The experiments are executed on a machine with an NVIDIA Tesla T4 GPU, 8 CPUs, 16GB of RAM, Keras 2.2.4 and TensorFlow 1.13.1.

4 Results

In this section the results of our experiments will be presented.

4.1 Domain Encoder

In this section, we will outline the performance of the model with and without the domain encoder.

Model	Accuracy	Recall	Precision	F1
W/ domains	0.8910	0.8914	0.8992	0.8953
W/o domains	0.8929	0.9143	0.8752	0.8943

Table 3: Model performance with and without the domain encoder

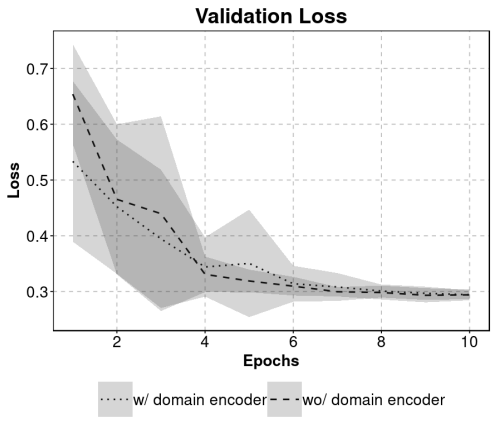
The performance impact of the domain encoder is outlined in Table 3. The results in the table are mean accuracy, recall, precision and F1 for 4 runs of 10 epochs for both configurations. As can be seen, the inclusion of the domain encoder does not seem to significantly impact performance, with the exception of slightly slower convergence as visualized in Figure 2. The shaded areas around the curves denote mean $\pm$ 1 standard deviation.

The model with the domain encoder has a smoother learning curve (fewer jumps) during the first couple of epochs. The spike in standard deviation around epoch 5 is caused by a single outlier run.

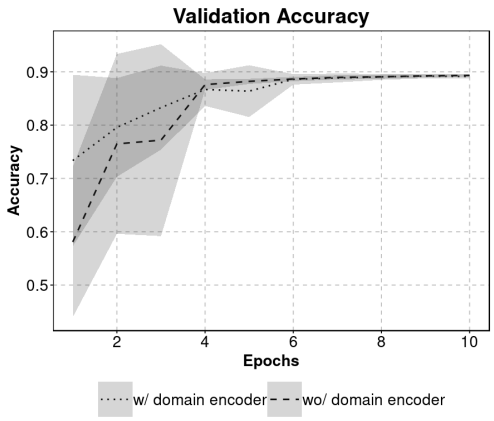
The accuracy and recall with the domain encoder are slightly worse, whereas the precision is slightly improved. Moreover, when looking at the F1 score we see a slight favor for using the domain encoder. This, in combination with the small differences, makes it hard to determine if performance is impacted either way. This will be discussed further in section 5.

4.2 Ablation Studies

During our research, we wanted to study the impact of ablating certain features of the re-

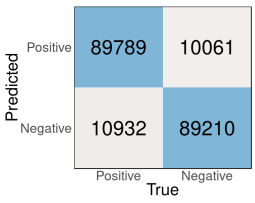


(a) The mean loss learning curves for the two configurations of the model

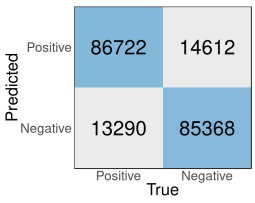


(b) The mean accuracy learning curves for the two configurations of the model

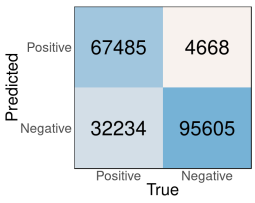
Figure 2: Loss and accuracy of the model with and without the domain encoder



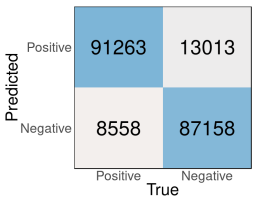
(a) Using all features



(b) Ablating adjectives



(c) Ablating stop words



(d) Ablating domains

Figure 3: Confusion matrices

view data. We wanted to do this to understand which word groups are particularly important for performing sentiment classification across multiple domains.

We hypothesized that adjectives and stop words were two important feature groups for determining sentiment and thus performed an ablation study on these.

The hypothesis for the stop words was based on the knowledge that the list of stop words used contains many negations such as 'not', which will distort the data. The reason for choosing adjectives was that certain descriptors of products might be negative or positive and that we may therefore see an impact in performance once those have been removed.

The domain encoder is included during the ablation experiments. The performance of a model in each of the studies plus a baseline, which is a model using all features of the data, can be found in tabular form in Table 4. These performance metrics are calculated on an unseen test set after training each model. For comparison the model without domain encoder is included as well.

Model	Loss	Accuracy
Using all features	0.2940	0.8910
Ablating stop words	0.4015	0.8185
Ablating adjectives	0.3803	0.8605
Without domain encoder	0.2943	0.8929

Table 4: Final performance on hidden test set in ablation studies

To further investigate these results, confusion matrices have been produced. They can be seen in Figure 3.

4.3 Domain Representations

The learned domain representations are extracted from the model and reduced in dimensionality using the PCA algorithm. To interpret the similarities between the representations, we employ hierarchical clustering analysis using ward linkage. In Figure 4 the result of the clustering analysis is displayed in a dendrogram.

5 Discussion

In this section, the implications of Section 4 will be discussed.

5.1 Domain Encoder

As discussed in Section 4.1, the performance does not seem to be greatly impacted by the addition of the domain encoder. In this section, we will attempt to highlight a few interesting observations about the architecture that could be potential points of further research.

Table 5 shows that the difference in accuracy is small. The accuracy after adding the domain encoder is slightly worse for almost all classes, but there does not appear to be a large inter-class difference in performance impact.

In Figure 5, we see the standard deviation of accuracy during training. The lower this value, the more consistent the model performs across runs.

What we see here is that the model with the domain encoder has a lower standard deviation initially. This suggests that the model is more consistent. Here we can also study the impact of the outlier observed in Section 4.1. We see clearly that the outlier has a large impact and that the rest of the runs are much more in agreement.

Domain	W/ domains	W/o domains
Automotive	0.8945	0.8957
Baby	0.9140	0.9140
Beauty	0.9041	0.9046
CD's and Vinyl	0.8534	0.8564
Cell Phones	0.9053	0.9079
Electronics	0.9024	0.9051
Personal Care	0.8931	0.8940
Movies and TV	0.8602	0.8635
Office Products	0.9013	0.9029
Sports	0.8943	0.8969
Toys and Games	0.8945	0.8941
Video Games	0.8792	0.8816

Table 5: Per domain accuracy performance with and without the domain encoder

Based on these results it could be argued that stability is increased, but without ensuring that the outlier observed is as rare as we assume, this cannot be concluded with any certainty. In addition, this stability seems to be inconsequential in this case as convergence is reached around the same time anyway.

5.2 Ablation Studies

Since we are developing a new model, and subsequently a new data set, we want to investigate whether or not all features (word groups in this case) of the data are important for sentiment classification across domains. In the case of this research, we theorized that both stop words and adjectives would be important for the performance. The aim of this part of the study is to verify some assumptions about the data set.

As shown in Table 4, the feature with the largest impact on performance is stop words. Negations are part of our stop word list. Therefore the distinction between classes will become less obvious for a model to see. The confusion matrix in Figure 3c shows that when stop words are removed the model predicts that positive samples are negative. In

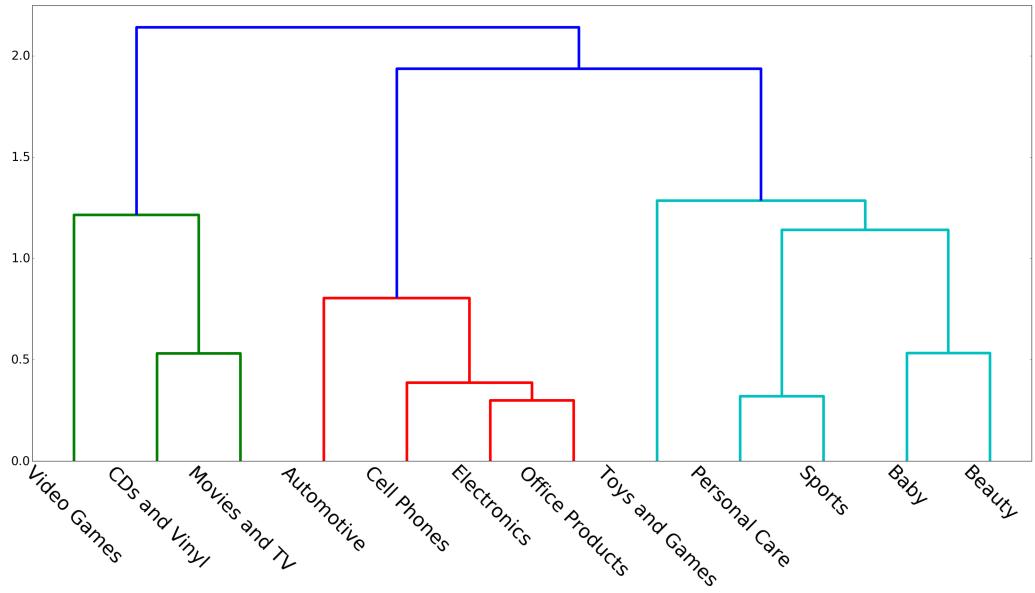


Figure 4: Dendrogram showing the clusters created by the HCA algorithm

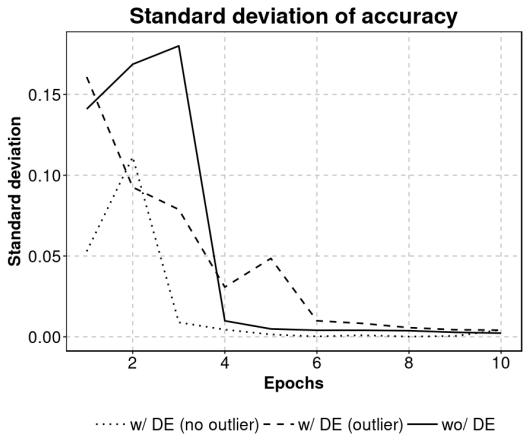


Figure 5: Standard deviation over time (DE = Domain Encoder)

fact, it predicts that 32% of all positive samples in the test set are negative. This is a major flaw with this model, as it predicts negative samples very well.

In Table 4, we see that ablating adjectives has less impact than ablating stop words. Also when looking at Figure 3b we see that ablating adjectives decreases overall performance thus not penalizing one class over the other.

These observations lead us to believe that our initial hypothesis of certain adjectives being used as a polarity marker, seems to not hold. However, the overall performance without adjectives is worse than with, so there might be some specific words that could be found during further research.

5.3 Domain Representations

As mentioned in Section 4.3, we use Hierarchical Cluster Analysis (HCA) to get insights into the similarities of the domain representations.

Doing this we get statistical insights into the similarities between the generated domain representations. These insights are presented in the dendrogram in Figure 4. The clustering provides us with three groups of domains that it finds the most similar. These three are the "green" (group 1), "red" (group 2) and "light blue" (group 3) groups.

Group 1 consists of digital media con-

tent domains: “Movies and TV”, “CD’s and Vinyls” and “Video Games”. It seems reasonable from a human point of view that these domains would be closely related. However, as can be seen in Figure 4, “Video Games” is the most distant domain in Group 1, merging with the others late in the clustering process.

Group 2 consists mainly of consumer electronics, with the odd one out being cars. It makes sense that cars is the least close of the bunch, but with the recent trend of adding a lot of electronics to cars, it is certainly not unreasonable that automotive reviews would share a lot of language with consumer electronics. Moreover, a lot of accessories, such as phone holders and chargers, are sold in the automotive category, leading to higher similarity.

Group 3 is definitely the most diverse of the three, containing seemingly unrelated domains like “Beauty” and “Toys and Games”. What the results suggest is that these categories may share some language characteristics at least in the context of determining the sentiment of the reviews. As we see though, the category of “Toys and Games” is less similar to any of the other domains within the group than the rest of the domains are to each other. However, it is more similar to the domains within its own group than to domains in other groups. We also see that the similarity between the two inner clusters of group 3 is small.

6 Conclusion

In the present work we have presented a model which represents domains as vectors and a way of understanding how the classifier is discriminating between different domains. The results seem to suggest that the model accurately captures the inter-domain relationships.

The model achieves performance comparable to that of a model without the proposed domain encoder, with hints at possible stability gains early in training.

Moreover, we find that stop words is an important feature group when performing sentiment classification. This is especially true when comparing to the importance of adjectives. While the performance does take a hit by removing the adjectives, it is not nearly as bad as with the stop words.

The convergence characteristics of the model would be an interesting point of further research. Moreover, looking into the use of the domain embeddings as a way of doing domain adaptation would also be a logical next step for this model.

7 Acknowledgement

We thank Barbara Plank for her guidance and advice. We would not have gotten this far without her.

References

Tanel Alumäe. 2013. Multi-domain neural network language model. In *INTERSPEECH-2013*, pages 2182–2186.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Autonomio. 2019. Talos. Retrieved from http://github.com/autonomio/talos.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain ner using cross-domain language modeling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 2464–2474.

Shou-Shan Li, Chu-Ren Huang, and Cheng-Qing Zong. 2011. Multi-domain sentiment classification with classifier combination. *Journal of Computer Science and Technology*, 26(1):25–33.

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 257–260, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qi Liu, Yue Zhang, and Jiangming Liu. 2018. Learning domain representation for multi-domain sentiment classification. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Hyeonseob Nam and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuanjing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2505–2513.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ottokar Tilk and Tanel Alumäe. 2014. Multi-domain neural network language model. In *Baltic HLT*, pages 149–152.