Testing Model Adaption Ability via Domain Shift on Movie Review Sentiment Analysis

SC4001 Project
Mohamed Ali Mohamed Umar - U2121706D
Shen Chihao - N2304724K
Phee Kian Ann - U2122217L

1 Introduction

This project is a study on Text Sentiment Analysis (TSA), which is the identification of emotions or opinions conveyed by the portion of text in question. It is a common classification task, typically resulting in either positive or negative outcomes. TSA has a wide variety of use cases that are commonly found in the business sector, and analysis of user feedback, such as reviews, can bring about valuable insights on the performance and success of certain products. With deep learning, we are able to analyse user-generated data automatically and thus be able to process a huge quantity of data that would have been impossible manually to garner more insights into user behaviour and feelings in the domain.

For this project, we are challenging the feasibility of domain adaptation so that a model that has been trained on one particular dataset can be re-used on multiple other datasets. The model is required to have flexibility to adapt to the introduction of new words, the absence of specific words, and possibly different phrasing contexts. The datasets used in this project are the SST-2 dataset (Socher et al., 2013) for the initial model training and testing and the IDMB review dataset (Maas et al., 2011) for the evaluation of our model performance for domain shift. We have experimented with three architectures of neural networks: Convolutional Neural Networks, Long-Short-Term Memory, and Hierarchical Attention Networks.

2 Existing Method

The application of sentiment analysis tools address domain shifting have advanced in a number of way. Using the collaborative LSTM-CNN architecture suggested by Marhapati et al. (2019) is one method. This design effectively adapts to domain-specific linguistic patterns and structural features found in review datasets by combining Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN). This model can effectively handle domain changes because the CNN is excellent at extracting local features, while the LSTM component captures long-term dependencies in sequential data. (Marhapati et al., 2019).

Another strategy is the CAN-CNN architecture put out by Xi et al. (2020), which shows a significant capacity to manage domain modifications in sentiment analysis with effectiveness. Using Category Attention Network (CAN), CAN-CNN is able to extract domain-specific category attribute words and dynamically modify attention weights (Xi et al., 2020).

Furthermore, Glorot et al. (2011) suggested using Stacked Denoising Autoencoders (SDAs) method for sentiment analysis tasks to handle domain shifts. Through unsupervised learning of high-level representations from unlabeled data, SDAs are able to identify underlying patterns and structures in the data (Glorot et al., 2011).

In summary, existing methods for handling domain shifts in sentiment analysis encompass a range of techniques, including collaborative architectures, attention mechanisms, and unsupervised learning approaches. These methods collectively aim to enhance model adaptability and performance across diverse datasets and domains.

3 Dataset Exploration

For this experiment, both SST-2 and IMDB are binary classification targets for positive and negative sentiments. The SST-2 dataset is split between train and test sets with 67349 and 872 rows of sentences, respectively, while we use the entire IMDB dataset as the evaluation set with a total of 50000 sentences.

4 Methods Implementation

4.1 CNN

Here, we try two different architectures of Convolutional Neural Networks. One uses word vectors as input, and the other utilises a character-wise embedding. We employ these two methods to demonstrate and fully discuss the robustness and generalisation ability of different embedding and CNN model structures.

4.1.1 Word Vector Attempt

Using word vectors derived from an unsupervised neural language model is a widely adopted technique to enhance performance, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Driven by the idea that words having comparable meanings frequently appear within comparable contexts, the former utilises either Continuous Bag-Of-Words approach or skip-gram to maximise the word similarity likelihood,

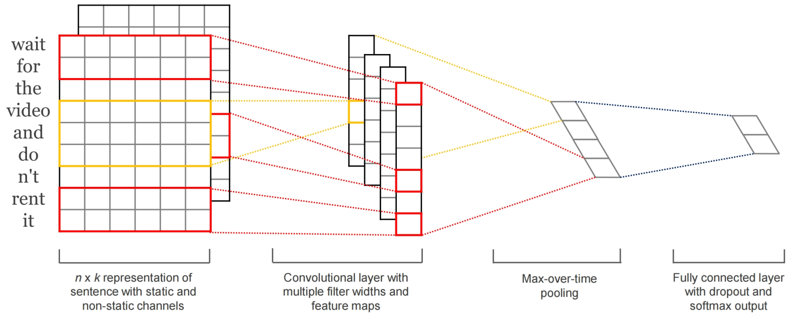
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: Our modified version of **CNN-WV**, using a combination of kernel size from 4 to 7.

while the latter employs a co-occurrence matrix to capture the substructure of word vector space.

To further explore the impact of various vector spaces on semantic classification and domain shift, we use both publicly given 300-dimensional word2vec vectors trained on a corpus of 100 billion words sourced from Google News and GloVe vectors pretrained on the Wikipedia 2014 and Gigaword 5 datasets in three different dimensions: 100, 200, and 300, separately. These word vector spaces are all used in the subsequent models featuring vector embedding, and thus we won't elaborate on them in the following sections.

For word vector attempt, we use a modified version of CNN architecture originated from Kim (2014) named **CNN-WV**. Different from the original model, which picks three kernels for the one-dimensional convolutional layer, we choose four different kernel sizes ranging from 4 to 7 with stride 1 and padding 5. The reason to choose these four kernel sizes is that they maximise the accuracy of the validation set and, at the same time, consider both short- and long-range correlations. After that, a maxpooling operation is applied to each feature map to obtain the maximum value, which is the most important feature. We concatenate all the maximum values from these three sets of maps to form a linear vector. A fully-connected softmax output is performed at the end to predict the probability distribution of each class.

For regularization, we directly use the methods from the original paper. We employ dropout on the final fc layer with a rate of 0.5 and an l_2 restriction by rescaling the Euclidean sum of all parameters of the fc layer to a certain number if it exceeds that number during each training loop.

4.1.2 Character-wise Attempt

As a contrast, we further employ an architecture with character-level embedding. We introduce the design from Zhang et al. (2015) named **CNN-Char**, which uses a one-hot implementation of 70 common characters, while the others are embedded with a zero vector.

Here we insert six convolutional layers with stride 1 at the bottom, along with three fully-connected layers at the top with a softmax output. We also add non-overlapping maxpooling layers for conv layers 1, 2, and 6 to capture the most important features. The model structure is shown in Table 1 and Table 2.

Layer	Feature map	Kernel	Pool
1	1024	7	3
2	1024	7	3
3	1024	3	N/A
4	1024	3	N/A
5	1024	3	N/A
6	1024	3	3

Table 1: Conv layers

Layer	Output Units
7	2048
8	2048
9	2048

Table 2: Fully-connected layers

We also add 2 dropout layers with a 0.5 probability in between the 3 fc layers for regularization. The only difference is that we limit the maximum length of input to 264 for simplicity since the maximum character length of the training set is 264.

4.2 LSTM

In this section, we attempt two implementations of LSTM models. The first implementation uses LSTM with multiple layers, and the second implementation uses self-attention mechanisms in addition to the LSTM layers.

Both implementations include a vectorisation process to transform textual data into a format that neural networks can process effectively. As mentioned above, we use different word vector spaces to convert words into dense vectors.

4.2.1 Bi-directional LSTM implementation

As can be seen in Table 3, we build a multi-layered LSTM model for the first implementation. The shape of the training data determines the input layer. The following layer is a Bidirectional LSTM, then a standard LSTM layer, then another Bidirectional layer, and finally, a dense output layer with a sigmoid activation function.

Layer	Type	Output Shape
1	Input	Determined by input size
2	Bidirectional	(Batch size, Sequence Length, 256)
3	LSTM	(Batch size, Sequence Length, 64)
4	Bidirectional	(Batch size, 256)
5	Dense	(Batch size, 1)

Table 3: Bidirectional LSTM structure

The choice of Bidirectional LSTM was driven by the need to extract bidirectional context information from the input sequences. The model can learn more about the context of each word in relation to its surrounding words by processing the sequences both forward and backward (Li et al., 2020). This is especially helpful in sentiment analysis, as the words that come before and after a word in a sentence can affect its sentiment.

Next, a standard LSTM layer is then used to extract additional characteristics from the sequential data. For computational efficiency, a single additional layer is selected, even though a more complex LSTM architectures have the possibility to capture more complicated patterns.

The final layer uses a single neuron with a sigmoid activation function, for binary classification task.

This implementation uses a simple architecture as it has fewer layers compared to the next implementation. This can be helpful in scenarios where computational resources are limited or when aiming for faster training times.

Layer	Type	Output Shape
1	Input	Determined by input size
2	Bidirectional	(Batch size, Sequence Length, 256)
3	Attention	(Batch size, Sequence Length, 256)
4	LSTM	(Batch size, Sequence Length, 64)
5	Attention	(Batch size, Sequence Length, 64)
4	Bidirectional	(Batch size, 256)
5	Dense	(Batch size, 1)

Table 4: Bidirectional LSTM - Attention structure

4.2.2 Bi-directional LSTM and Attention implementation

Table 4 illustrates how we added self-attention layers to the model architecture based on the prior approach, giving the model the capacity to concentrate on significant segments of the input sequence. The layers were implemented with RelU activation function.

In order to enable the model to learn attention weights for every word in the sequence, the self-attention layers are added after the LSTM layers. The attention mechanism allows the model to focus more on words that are more useful for sentiment categorisation by assigning words a range of importance rating (Basiri et al., 2021).

The goal of this approach is to extract more intricate patterns and dependencies from the data by using more LSTM layers and self-attention processes.

4.3 HAN

The objective of Hierarchical Attention Networks (HAN) for text classification tasks is to capture the hierarchical structure of documents, where sentences are sets of sequenced words and documents are sets of sentences (Yang et al., 2016). The general structure of HAN, as shown in figure 2, depicts the general structure of HAN, which is made up of two neural network levels, beginning at the word level with a word encoder to word attention layer, which is then fed into the sentence level, sentence encoder to sentence attention layer, and finally the output layer. HAN brings about advantages over traditional single sequential approaches, allowing us to capture both word-based local information and sentence-based global information, which would allow the model to better predict some cases of classification that might be hard otherwise.

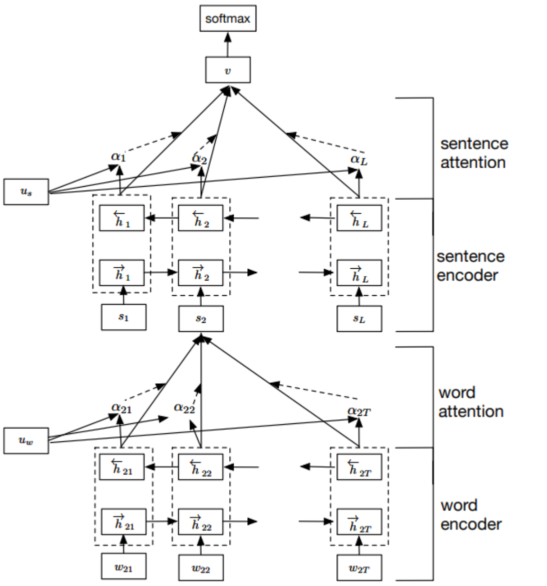

Figure 2: Overall architecture of HAN (Yang et al., 2016)

4.3.1 HAN implementation

For our implementation of HAN, as illustrated in Table 5, we use two Bidirectional LSTMs with an output dimension of 100x2 each and a dropout value of 0.3 as the base models while applying the HAN architecture with softmax activation for the output to it. The model uses Root Mean Square Propagation as the optimisation algorithm, with minimising categorical cross-entropy loss as its target. Input data is also tokenized and weighted using word2vec vocabulary in a 3D matrix of words, sentences, and token weights rather than a 2D matrix in Sections 4.1 and 4.2.

Word Layer

Layer	Type	Output Shape
1	Input	(Batch size, 50)
2	Embedding	(Batch size, 50, W2V Dim)
3	Bidirectional (LSTM)	(Batch size, 50, 200)
4	Attention	[(Batch size, 200), (Batch size, 50, 1)]

Sentence Layer

Layer	Type	Output Shape
1	Input	(Batch size, 50, 50)
2	Time Distributed	(Batch size, 50, 200)
3	Bidirectional (LSTM)	(Batch size, 50, 200)
4	Attention	[(Batch size, 200), (Batch size, 50, 1)]
5	Dense	(Batch size, 2)

Table 5: HAN structure

5 Experimental and Evaluation

5.1 Experiment Setup

In order to discover the effect of domain adaptation, we apply two datasets with the same semantic classification task.

Our task focuses on movie review Text Sentiment Analysis (TSA). Given movie reviews written by different users, we are able to classify the sentiment expressed in the review as positive or negative. This involves analysing the text to understand the underlying emotions and opinions conveyed by the reviewer. We use the SST-2 dataset as our training, validation, and testing set in the first place. Then we use the IMDB dataset for domain shift testing in order to test the generalisation performance of different models.

For those that need hyper-parameter tuning, We divide the SST-2 training set into 0.8 for training and 0.2 for validation and fine-tune the model based on batch size and feature map number using word2vec vector space. For **CNN-WV**, we choose the feature map number to be 256 for each kernel size and use mini-batch gradient descent of size 128 via grid search. By some empirical attempts, we also set the learning rate to 0.0005 and the l_2 constraint to 3 for regularization. For **CNN-Char**, we directly use the hyper-parameters offered by the author with a mini-batch size of 128. We set a learning rate of 0.0001 for convergence. For the above two CNN models, we also set a patience of 10 for early-stopping. For **LSTM**, we set the batch size to 128 and the learning rate to 0.001 and implemented early-stopping with a patience of 5, and for **LSTM-Attention**, we used the same configuration other than to set the optimised mini-batch size to 64. For **HAN**, we set the optimised mini-batch size to 64 and the learning rate to 0.001 while employing early-stopping with a patience of 5.

We test the models (except **CNN-Char**) on different word vector spaces, namely word2vec and GloVe for SST-2. But for IMDB, we only use word2vec vector space since it has the best performance on average.

5.2 Stop Words

We further discuss whether text pre-processing via the removal of stop words, which are common words considered insignificant for NLP tasks such as "a", "the", "is", and are (Brigadir and Nothman, 2019), would enhance models' performance and generalisation ability on both the original test set and the IMDB dataset. Apart from common stop words, we also remove other unnecessary components such as emails, urls, html tags, and accented characters. Additionally, we apply context expansion to convert contracted words into their formal forms (e.g., thats -> that is). We use the `text_hammer` package in Python to perform this task. Table 6 illustrates the difference in words with or without stop word removal.

Dataset	Original	Non-stop	Word Loss
SST2-Train	583810	338193	~42%
SST2-Test	15133	8419	~44%
IMDB	11468566	6049490	~47%

Table 6: Word count of datasets with or without removal of stop words

5.3 Evaluation and Result

Table 7 shows the test result on the SST-2 dataset. The best models of different vector spaces all reach an accuracy of around 0.85, except the one-hot character space, which has the worst performance. While **CNN-WV** outperforms on both word2vec-300 and GloVe-200, **LSTM-Attention** has a better performance on GloVe-300, and **HAN** exhibits superior performance with GloVe-100. There is no clear evidence showing that larger-dimensional vectors contribute to higher accuracy. However, it is definite that eliminating stop words is not advisable for all models on the sentiment classification task, as the accuracy drops by about 3 to 6 percent.

Model	word2vec-300	GloVe-100	GloVe-200	GloVe-300	Char
CNN-WV	**0.870**	0.839	**0.839**	0.842	-
CNN-WV (Non-stop)	0.819	0.800	0.791	0.802	-
CNN-Char	-	-	-	-	**0.743**
CNN-Char (Non-stop)	-	-	-	-	0.702
LSTM	0.845	0.835	0.832	0.799	-
LSTM (Non-stop)	0.810	0.778	0.786	0.778	-
LSTM-Attention	0.849	0.844	0.841	**0.851**	-
LSTM-Attention (Non-stop)	0.800	0.818	0.802	0.810	-
HAN	0.797	**0.853**	0.835	0.844	-
HAN (Non-stop)	0.773	0.794	0.798	0.784	-

Table 7: Test accuracies on SST-2. Non-stop means removing the stop words for both training and testing data. The best model is our modified version **CNN-WV**, which has an accuracy around 87%. Our **HAN** also shows a good performance of 85.3% accuracy on **GloVe-100**.

Model	IMDB
CNN-WV	**0.838**
CNN-Char	0.530
LSTM	0.742
LSTM-Attention	0.822
HAN	0.797

Table 8: Test accuracies on IMDB

Table 8 lists the outcome on the IMDB set. **HAN** performs decently but leaves more to be desired compared to the other higher performing models. While **CNN-WV** and **LSTM-Attention** exhibit relatively strong capability for generalisation, the rest seem to be not robust enough on the domain shift. **CNN-Char** even has such a terrible accuracy of 53% that it is almost comparable to random guesses.

6 Discussion

6.1 Domain and Setup

Figure 3 displays t-SNE visualisation (van der Maaten and Hinton, 2008) of the superficial semantic features of two datasets. Both datasets have a similar mean value. Though some points of different datasets mix together in either plot, it is obvious that the IMDB set has a larger variance, resulting in a wider range of diversity, while a certain amount of data in SST-2 is concentrated at its core. Therefore, though the task is similar, the domain difference can adequately demonstrate if a model has sufficient generalisation ability or not.

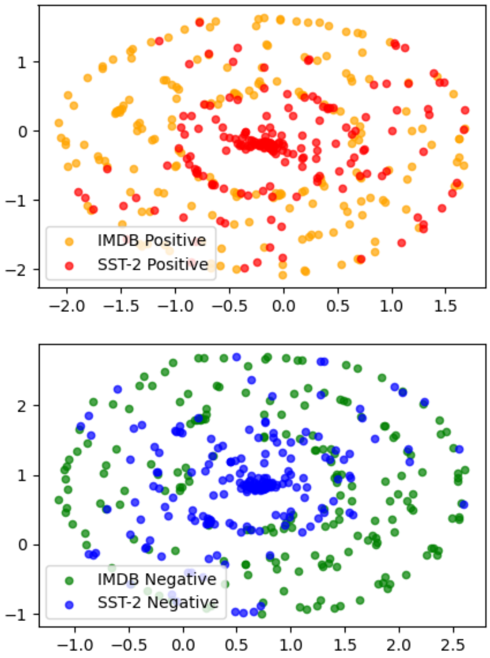

Figure 3: t-SNE visualization of superficial semantic features. We randomly sample 200 positive and 200 negative data points from both datasets, encode each into a word2vec embedding and use t-SNE to visualize in a two-dimensional space.

We expected that removing stop words might yield better outcomes, as the remaining words are likely to be more meaningful and hold more valuable information. However, it turns out that the results get even worse for all models. With a total word count loss of more than 40% for all datasets, we think it may be due to the possible side effects that it brings about, such as loss of sentence structures, missing word meaning, and altering contexts. It might have led to semantic change and caused the model to fail to detect certain phrase patterns within a specific range. Thus, it is not recommended to eliminate stop words in a semantic analysis task such as user reviews.

Meanwhile, there is no conclusive evidence indicating a direct link between the performance of the model and the embedding dimension. As seen in Table 7, even the least embedded dimensional word vector space GloVe-100 exhibits better accuracy than both 200 and 300 dimensional GloVe spaces for models like **LSTM**, **LSTM (non-stop), and HAN**. This may be due to the fact that a lower-dimensional embedding space could capture more essential semantic information while reducing noise and overfitting. However, the optimal embedding dimension might vary depending on the specific model and dataset characteristics. Further experimentation and analysis are necessary to fully understand the relationship between embedding dimension and model performance.

6.2 Model Performance

CNN-WV shows a superior result on both the original dataset and the domain shift challenge. It seems Convolutional layers are quite adapted to gathering meaningful features for word vectors. Though the structure is simple, different kernel sizes capture information from different ranges, which is fair enough. And since the word length of each data point is relatively short, the kernel sizes are adequate for the dataset. In the meantime, by embedding words into dense vector representations, CNN-WV can effectively capture the contextual relationships between words, enhancing its ability to perceive subtle differences in meaning.

On the other hand, it is struggling for **CNN-Char** to have similar performance. Though using 6 Conv layers, 1024 feature maps, and an even more complex architecture, characters seem not to be strong enough to capture the underlying semantic connections. Character-wise embedding also leads to unnecessary behaviour since a certain character might appear lots of times in a sentence, which causes the model to build redundant bonds and become overfitting to these bonds. Meanwhile, a kernel size of 7 on the shallow layers may also restrict the model's performance, as it may only capture information at the word level and is not able to capture features from different ranges. A limitation of 264 character length also contributes to the bad result since the model is not able to see the whole data point for the test set during domain shift.

While both the **LSTM** and **LSTM-Attention** models exhibit strong accuracy during training, the **LSTM-Attention** model, with its integration of self-attention mechanisms alongside bidirectional LSTMs, demonstrated superior resilience to domain shifts, as evidenced by its higher accuracy during testing compared to the regular **LSTM** model. The self-attention layers in the **LSTM-Attention** model likely facilitate a more nuanced understanding of the input sequences, enabling it to capture domain-independent features more effectively.

In contrast, the regular **LSTM** model did not do as well during testing. This might be due to the lack of attentional focus that is present in the **LSTM-Attention** model, which likely results in a limited ability to adapt to the nuances present in the testing dataset, leading to a notable drop in accuracy in testing. As a result, the regular **LSTM** model struggles to generalise effectively across domains.

Although the hierarchical structure of HAN seems to be ideal for this domain shift challenge due to its ability to capture more information, on the contrary, it did relatively worse than a simple implementation of LSTM and CNN. This was a surprising outcome and might be due to the fact that reviews are far too short in word length. Thus, the benefits of distinguishing between words and sentences did not provide too many beneficial insights and were even further detrimental to model performance. There was simply too much processing done for these datasets, which led to increased classification noise and model performance degradation. It is possible that HAN would perform better on different datasets, such as summaries of articles or novels.

7 Conclusion

In this study, we test our deep learning models, including CNN, LSTM, and HAN, on sentiment analysis tasks across different domains and explore their generalisation abilities. Through experiments on the SST-2 dataset and evaluation on the IMDB dataset, we assess the models' robustness and adaptation capabilities in dealing with domain shift. Our results revealed that different models obtain a certain amount of domain adaptation ability. We explain our findings about the setup methods and also discuss the reasons why some models perform better and others perform worse. Further research can be done with more diverse amounts of datasets in order to robustly test model capabilities, and fine-tuning of models can be considered an improvement to this project. Overall, our discoveries contribute to a deeper understanding of the domain adaptation abilities of different models and provide insights for future research.

References

Mohammad Ehsan Basiri, Shahla Nemati, Moloud Abdar, Erik Cambria, and U. Rajendra Acharya. 2021. Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems*, 115:279–294.

Igor Brigadir and Joel Nothman. 2019. igorbrigadir/stopwords: First release.

Xavier Glorot, Antoine Bordes, and Y. Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Weijiang Li, Fang Qi, Ming Tang, and Zhengtao Yu. 2020. Bidirectional lstm with self-attention mechanism and multi-channel features for sentiment classification. *Neurocomputing*, 387:63–77.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Savitha Marhapati, Ayesha Nafeesa, R. Tanuja, S.H. Manjula, and K.R. Venugopal. 2019. Semi-supervised domain adaptation and collaborative deep learning for dual sentiment analysis. In *SN Applied Sciences*. Springer Nature.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

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, Doha, Qatar. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Dongbo Xi, Fuzhen Zhuang, Ganbin Zhou, Xiaohu Cheng, Fen Lin, and Qing He. 2020. Domain adaptation with category attention network for deep sentiment analysis. pages 3133–3139.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649657, Cambridge, MA, USA. MIT Press.