# A Shot in the Dark:
# Modeling Improved Zero-Shot and Few-Shot Transfer Learning with Self-Supervised Models for Sentiment Classification

**Project Category: Natural Language**
**Project Mentor: Proposal Graded by Ananya Kumar**

**Name: Anwesha Mukherjee**
SUNet ID: anwesham
Department of Computer Science
Stanford University
`anwesham@stanford.edu`

**Name: Olivia Lee**
SUNet ID: oliviayl
Department of Computer Science
Stanford University
`oliviayl@stanford.edu`

**Name: Raj Palleti**
SUNet ID: rpalleti
Department of Computer Science
Stanford University
`rpalleti@stanford.edu`

## 1 Key Information to include

Anwesha and Raj submitted a related project for CS230 which shares the data preprocessing and we take results from the DistilBERT experiements.

## 2 Introduction

Large language models, or more recently coined foundation models, have become increasingly prevalent in AI [1]. There has been a growth in the application of transfer learning techniques to these models, which refers to the ability to fine-tune a transformer model for one task and adapt it's use to another. This helps solve the small data problem, where some domains may lack processed data to directly tune a model, and thus, finding a similar frame of data, can be used to adapt to the domain in what's known as domain adaptation. Such transfer learning tasks have been successfully demonstrated on BERT, especially for cross-lingual transfer [2][3][4]. However, previous investigations have revealed that deep models struggle with up-to-par performance in transfer for sentiment analysis without similar corpuses in context [5] [6]. Therefore, we aimed to see if we can effectively model transfer learning without attention, testing different machine learning, deep learning, and self-supervised techniques to optimize performance on binary sentiment classification and compare to DistilBERT, a portable BERT model. We wanted to definitively determine if scale and attention mechanisms in current language models truly offer a clear advantage for transfer, the purpose of their development. We use various binary sentiment classification datasets text inputs and binary label outputs (positive or negative, processed according to model) with variability tested on size, vocabulary size, difference in domain, and polarity to better understand influences on transfer capabilities.

## 3 Related Work

Traditionally, sentiment analysis can use lexicon-based approaches which judge sentiment polarity by constructing an emotional dictionary, extracting emotional values from inputs. Proposals included using sentiment polarity to analyze the tendency of text sequences for a sentiment [7], generating a dictionary of positive and negative sentiment words from seed words in WordNet which banks the most common words for various mathematical relationships that determine positive or negative sentiment [8], using supervised learning to form an opinion lexicon [9], and a SO-PMI function-based algorithm calculates semantic orientation of an input based on positive and negative-leaning words [10]. However, the lexicon approach relies entirely on a sentiment dictionary, which ignores positional connections and does not learn on any relational terms including different conjugations of a word. Other baselines standardly use Naive Bayes and SVM, but these two don't have any framework to operate beyond zero-shot transfer without a validation scheme and rely on exact vocabularies [5].

Previous surveys of sentiment analysis domain adaptation found that the gap in domain material poses a major challenge for positive transfer. As an example, a model trained by a film review corpus will have poor performance when used to analyze restaurant reviews due to excessive difference between the domains [5]. Other past work even found examples of negative transfer when input and output data wasn't similar enough with complete sequential learning that simply trains upon an existing dataset [6]. However, these researchers further suggested focusing on three areas of transfer: parameter, instance, and feature representation [5]. Parameter transfer involves leveraging the parameter sharing model of the source and target domains, transferring the trained model parameters in large number of datasets to the target task, which we adopted in this project. Feature transfer representation is used when source target domains have part crossover features. One can transform data from the two domains into the same feature space and perform traditional machine learning. We drew inspiration from feature representation transfer in our use of DistilBERT and word2vec.

## 4 Dataset and Features ≈ 0.5-1 pages

We used the following 4 datasets for evaluating the performance of transfer learning: 1) 50K IMDb Movie Reviews, 2) 9K tech product review tweets, 3) 50K general domain polar tweets, and 4) 50K Rotten Tomatoes movie reviews. For the IMDB, Rotten Tomatoes, and Social Twitter datasets, we used an 80-10-10 split for training, validation, and testing. After processing, our data included an input string and a label (either positive or negative sentiment).

| | |
|---|---|
| I thought this was a wonderful way to spend time | positive |
| This movie was a total waste of time. Hated it. | negative |

Additionally, while sampling, we ensured we'd have reproducible and identical few-shot samples across models by using the same random state. We sampled to ensure we used less than 20% of available data, in 3/4 cases 5% of available data would be augmented atop training data.

For feature testing, we developed trained embedding matrices for an embedding layer using word2vec with the continuous bag of words (CBOW) framework. CBOW predicts a center word from surrounding context in terms of word vectors with a feed-forward, window-limited neural network (here we use window size 7), given tweet parsed (300 max. length) inputs after processing for the LSTM. CBOW word2vec constructs an input word vector: $\mathcal{V} \in \mathbb{R}^{n \times |V|}$, and an output word vector: $\mathcal{U} \in \mathbb{R}^{|V| \times n}$, that are optimized using stochastic gradient descent:

$$\begin{aligned}
\underset{u_c, v_j}{\operatorname{argmin}} J &= -\log P\left(w_c \mid w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}\right) \\
&= -\log P\left(u_c \mid \hat{v}\right) \\
&= -\log \frac{\exp\left(u_c^T \hat{v}\right)}{\sum_{j=1}^{|V|} \exp\left(u_j^T \hat{v}\right)} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp\left(u_j^T \hat{v}\right)
\end{aligned}$$

based on the cross entropy loss function $H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log\left(\hat{y}_j\right)$ [11]. This pre-establishes an embedding matrix for the embedding layer that can be frozen.

## 5 Methods

We strived to test a machine learning algorithm with integrated validation, a deep learning algorithm (without a vanishing gradient risk), and a deep learning algorithm with a self-supervised learning component. Thus, we tested implementing a validation scheme within logistic regression, an LSTM, and an LSTM with word2vec initialized embeddings.

### 5.1 Logistic Regression

We implement the standard logistic regression model with gradient descent on $\theta$ using the training set, with the addition that in every iteration, we calculate the accuracy metric on the validation set and decreased the learning rate (multiplying by a factor of 0.9) if the validation accuracy was decreasing. This allowed us to effectively check for overfitting to the source domain in transfer learning with controlled learning based on the target data for gradient management and smoothen out learning to better approach a validation loss minimum. We also had an additional early stopping condition if the change in validation accuracy is within a small threshold ($\epsilon = 1 \times 10^{-5}$) to further avoid overfitting.

### 5.2 LSTM

In our LSTM model we have the inputs converted into their embeddings with sparse dropout before an LSTM layer of 100 cells (downsize dimensions by 3 from embeddings with width of 300) 1.

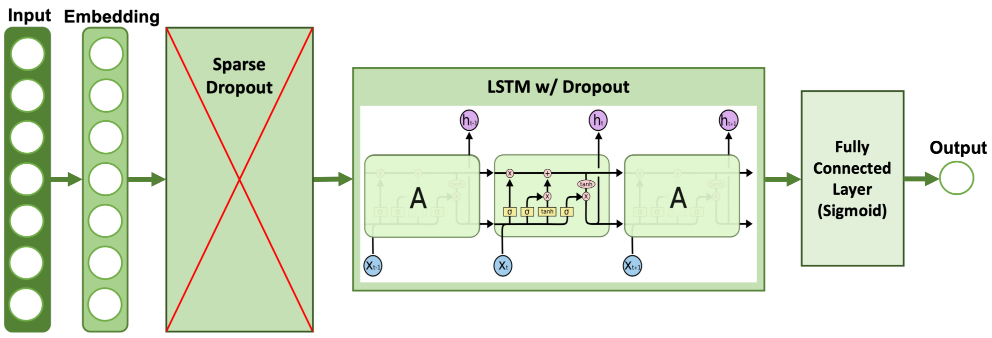
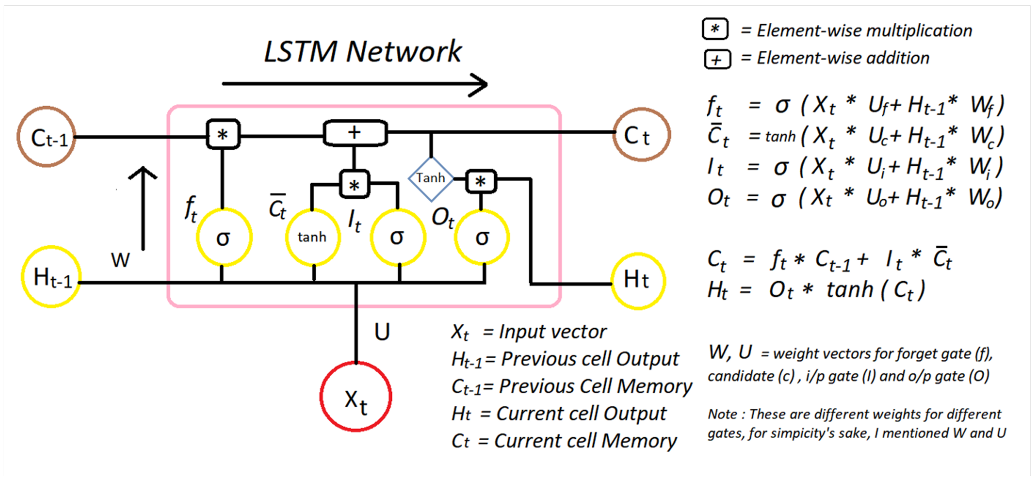


Figure 1: Diagram detailing the architecture of the LSTM employed

Zooming in on the individual cells of the LSTM, we can see the forget gate ($f$), candidate layer ($\bar{C}$), the input gate ($I$), and output gate ($O$) at a given timestep as single-layer feed-forward neural networks with sigmoid activation, with the exception of the candidate layer which uses a tanh activation function. At any stage of the LSTM, inputs are the current input: $\mathbf{X}_t$, the previous hidden state: $\mathbf{H}_{t-1}$, and the previous memory state: $\mathbf{C}_{t-1}$ to output a the current timestep's hidden: $\mathbf{H}_t$ and memory: $\mathbf{C}_t$ states 2. The element-wise multiplication with the forget gate allows important gradients to be retained, and unimportant gradients to be passed without impact which avoids the vanishing gradient problem.



LSTM Network
= Element-wise multiplication
= Element-wise addition
$f_t = \sigma(X_t * U_f + H_{t-1} * W_f)$
$\bar{C}_t = \tanh(X_t * U_c + H_{t-1} * W_c)$
$I_t = \sigma(X_t * U_i + H_{t-1} * W_i)$
$O_t = \sigma(X_t * U_o + H_{t-1} * W_o)$
$C_t = f_t * C_{t-1} + I_t * \bar{C}_t$
$H_t = O_t * \tanh(C_t)$
$X_t$ = Input vector
$H_{t-1}$ = Previous cell Output
$C_{t-1}$ = Previous Cell Memory
$H_t$ = Current cell Output
$C_t$ = Current cell Memory
W, U = weight vectors for forget gate (f), candidate (c), i/p gate (I) and o/p gate (O)
Note : These are different weights for different gates, for simpicity's sake, I mentioned W and U

Figure 2: Single cell of an LSTM [12]

## 6 Experiments/Results/Discussion

Given the task was a binary sentiment classification task without an objective priority for one label over the other, accuracy served as our primary metric, though we generated confusion matrices for the LSTMs and evaluated precision, recall, and F1 to better understand variance in learning performance. All graphs, focus on accuracy, as it was ultimately the most representative for primarily evenly distributed and valued data in opposition to a task like Named Entity Recognition where the f1-score is most reflective.

$$\begin{aligned}
\text{Accuracy} &= \frac{TP+TN}{TP+TN+FP+FN} \\
\text{Precision} &= \frac{TP}{TP+FP} \\
\text{Recall} &= \frac{TP}{TP+FN} \\
\text{F1-score} &= \frac{2 \times \text{ Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}}
\end{aligned}$$

### 6.1 Logistic Regression with Validation

Logistic Regression with validation tends to perform more poorly than other models. However, it notably transfers much better to smaller target datasets, like the Twitter Product dataset. In general, performing transfer learning from a larger to a smaller dataset usually works better than the converse.

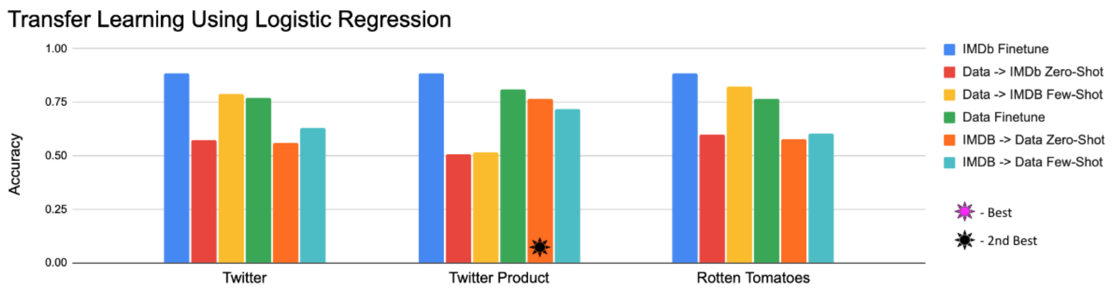


Figure 3: Accuracy values for transfer learning using logistic regression with validation

### 6.2 LSTM

LSTMs perform well in transfer learning for most tasks, almost always outperforming logistic regression. The LSTM outperforms all other models in fine-tuning for the Twitter Product dataset, which is the smallest dataset, indicating that LSTMs may work better than large deep learning models if the dataset is small. Performance particularly improved on transfer, likely due to better contextual learning.

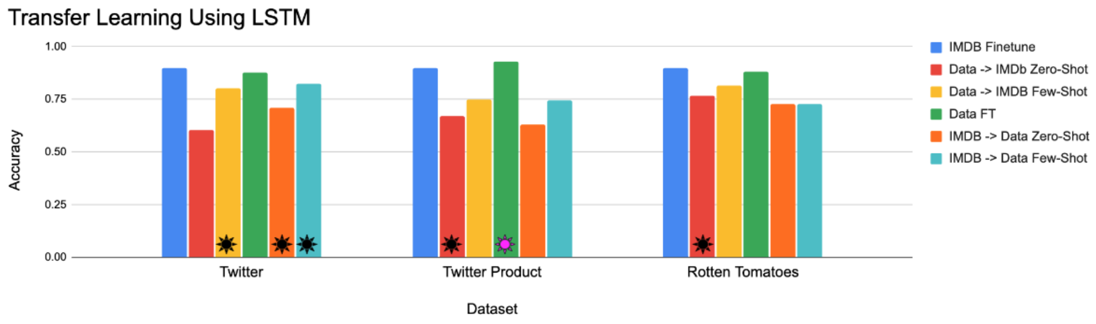


Figure 4: Accuracy values for transfer learning using LSTM

#### 6.2.1 Architectural Decisions

**Sigmoid Activation:** Unlike most deep learning models, because of the forget gate in an LSTM handling the vanishing gradient, sigmoid functions are often more balanced. In testing for finetuning on both IMDb and Twitter Product data, the sigmoid function outperformed GeLU, tanh, and ReLU activation functions, so we chose to maintain it. This made sense given the structure of sentiment analysis, which operates on a binary that should have polarity in both directions for 0 to 1 scaling.

**Sparse Dropout:** In order to avoid embedding training that is non-contextual, we opted to use sparse dropout rather than direct dropout due to the realization that the model was retaining randomized gradients which could muddle information, especially when word2vec initialized. We still maintained recurrent dropout within the LSTM for weight adaptation to account for individual relationships without negatively altering the embedding layer in training.

**Decay and Early Stopping:** After initial testing, we determined that convergence would always occur prior to 32 epochs, but often, the best model would be passed over and the gradient wouldn't be able to centralize. Thus, we added in decay according to validation loss by testing for plateau, maintaining patience of 2 epochs to ensure the gradient was in-fact overshooting by a scale of 10. In the future, it may be valuable to test epoch-based decay with patience. Additionally, if the validation accuracy didn't improve after 5 epochs (patience = 5), then we utilized early stopping callback as it appeared the model had reached either convergence or some form

#### 6.2.2 Different Embedding Structures

Similar to DistilBERT where we found better performance when attention layers were also tuned, we decided to try different patterns of embedding the inputs and testing if frozen pretrained word embeddings would help or hurt performance as a self-supervised component.

**Trainable Zero Intialization 4:** Operating on the standard LSTM, we initialized zero-vector embeddings for an empty embedding matrix that was only modified. The model was the most adaptable, though it frequently took more epochs to converge particularly for transfer.

**Pretrained word2vec 5:** When freezing the embedding layer, much of the model's adaptability was lost with firm input schemes entering the LSTM reducing the impacts of the gradient in learning. This led to poorer convergence in many cases since the model was being simplified, though in the case of Twitter to IMDb zero-shot learning, there was substantial improvement due to large vocabulary variance of the twitter sentiment dataset which encodes many of the necessary word associations for imdb data.

**Trainable word2vec 6:** Similar to BERT, we then further considered word2vec initialized matrices that are optimized to account for new words and associations as a result of the validation data. This, however, can have adverse effects because associations originally learned can be altered when training. In this case, it seems the trainable word2vec is particularly a better initialization when corpora are similar as with Rotten Tomatoes and IMDb data.

### 6.3 DistilBERT

Operating as an oracle, using results from CS230, the DistilBERT was found to be optimized for fewshot learning with default parameters. In testing the design of the LSTM we tried to match parameters, but found that different parameters worked best, but even so, DistilBERT outperformed the LSTM on all but 1 out of 16 metrics (IMDb finetune repeats 3 times on graph for uniformity).

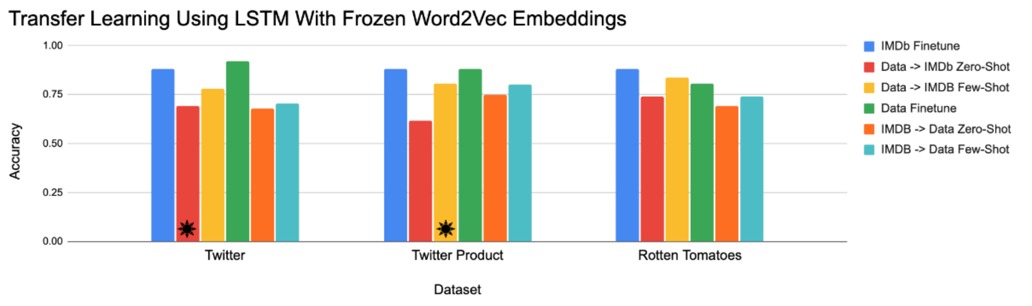


Figure 5: Accuracy values for transfer learning using LSTM with frozen word2vec embeddings

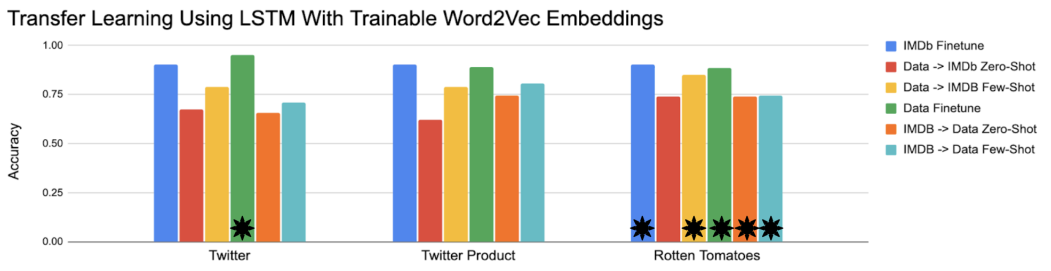


Figure 6: Accuracy values for transfer learning using LSTM with trainable word2vec embeddings

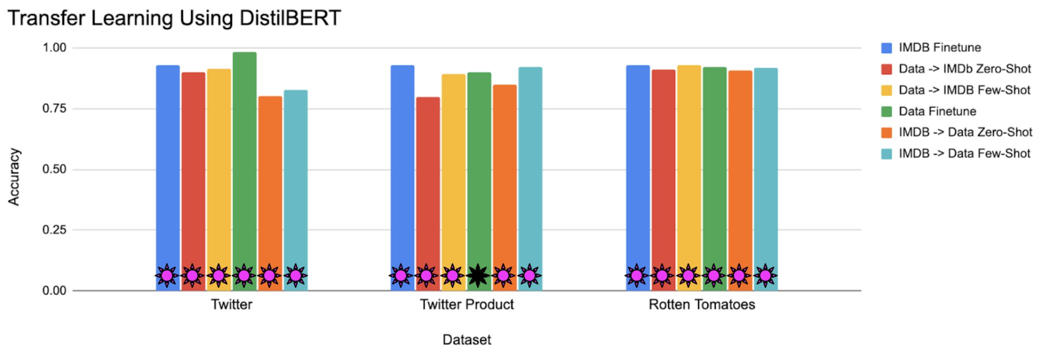


Figure 7: Accuracy values for transfer learning using DistilBERT

## 7 Conclusion/Future Work

DistilBERT outperformed all other models for most tasks, demonstrating the utility of large language models for transfer learning. Pretrained attention schemes likely play large role in success of DistilBERT. We find that transfer learning works better when transferring from a larger dataset to a smaller dataset, and that having a larger vocabulary is generally better for transfer. Surprisingly, data polarity didn't play an important role in comparison to vocabulary variance and size. From our experiments involving word2vec embeddings, we find that frozen word2vec embeddings severely limit model adaptability, and using zero-initialized or word2vec trainable embeddings performs better in different contexts. This is because there is a risk of overfitting to the training set when using trainable word2vec embeddings, especially if the train data and test data are selected from different contexts. We also find that using trained embeddings from word2vec often reached convergence quicker than zero initialization regardless of performance. This is due to immediate gradient impact and closer positioning to minimum loss. Finally, for logistic regression, we find that the lack of a validation gradient hurts performance.

Future research may include testing transfer learning on Rotten Tomatoes with frozen word2vec embeddings, as zero-shot transfer improved by 4% in accuracy when augmenting the text8 (sampled Wikipedia data) corpus, which simulates pretrained embeddings. We may also further experiment with different activation functions (e.g. GeLU, ReLU) within the LSTM cells rather than the fully connected layer to match DistilBERT and sharpen prediction boundaries. Finally, we could try stacking LSTMs to form a more portable attention model.

## 8 Contributions

In code, Anwesha worked on the data preprocessing, Anwesha and Raj worked through the LSTM models architecture, where Raj conducted tests before Anwesha added Word2Vec (with frozen and trainable word embeddings) to conduct additional tests. Olivia worked on the code for transfer learning for logistic regression. Raj constructed logistic regression and MLP baselines, Olivia worked on SVM, and Anwesha worked on Naive Bayes. Anwesha and Raj worked on the notebooks for transfer learning using DistilBERT, which was done as a part of the CS 230 project. All members ran their code for determining the performance of transfer between each pair of datasets, in both the zero-shot and the few-shot setting. All members contributed to the report and poster.

## References

[1] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics.

[4] Yuki Arase and Jun'ichi Tsujii. Transfer fine-tuning: A BERT case study. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5393–5404, Hong Kong, China, November 2019. Association for Computational Linguistics.

[5] Ruijun Liu, Yuqian Shi, Changjiang Ji, and Ming Jia. A survey of sentiment analysis based on transfer learning. *IEEE Access*, 7:85401–85412, 2019.

[6] Ariadna de Arriba, Marc Oriol, and Xavier Franch. Applying transfer learning to sentiment analysis in social media. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 342–348, 2021.

[7] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, oct 2003.

[8] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA, 2004. Association for Computing Machinery.

[9] Felipe Bravo-Marquez, Eibe Frank, and Bernhard Pfahringer. From opinion lexicons to sentiment classification of tweets and vice versa: A transfer learning approach. 10 2016.

[10] Ai Yang, Jianghao Lin, Yong Zhou, and Chen Jin. Research on building a chinese sentiment lexicon based on so-pmi. *Applied Mechanics and Materials*, 263-266:1688–1693, 12 2012.

[11] Christopher Manning and Richard Socher. Cs224n: Natural language processing with deep learning lecture notes - word vectors i: Introduction, svd and word2vec, January 2019.

[12] Tim O'Shea, Seth Hitefield, and Johnathan Corgan. End-to-end radio traffic sequence recognition with deep recurrent neural networks. 10 2016.