Sentiment Classification on Reddit comments on Climate Change

Jaskirat Marar
732A81 - Text Mining
jasma356@student.liu.se

Abstract

This project takes the form of an investigative study in analyzing user comments posted on Reddit to determine the user's sentiment on Climate Change. The study proposed to compare the classification capabilities of simple RNNs to learn semantics & syntactic form of the comments versus other variants like LSTMs and GRUs that capture the long-term dependencies using word embeddings from pre-trained GloVe vectors. The importance of context is showcased clearly when we combine word embeddings from GloVe with a LSTM which results in a boost in the quality of results with a f1 score of 75% vs 72% f1 score for a simple RNN. We attempt to improve upon the performance of the simple LSTM by replacing it with a deep GRU architecture and use dropout and pre-trained word embeddings from GloVe to receive a f1 score of 77%.

1 Introduction

Sentiment Analysis as a research field has over really benefitted from advancements in Machine Learning and better access to computational resources. Before all the modern algorithms and tools available to researchers today, some of the earliest methods used in sentoment analysis were Rule-based approaches which relied on manual rules to identify sentiment present in a given text. Turney (2002) showed that this was difficult to scale for large datasets and did not always capture context and nuance of the text. With the rapid advancements in Machine Learning, statistical modelling became the choice solutions for sentiment analysis. Supervised learning algorithms such as Support Vector Machines (SVMs) and Naive Bayes were shown by Pang and Lee (2008) to achieve high accuracy in sentiment classification tasks. This was followed by the growth in popularity of Deep Learning which led to wider use of CNNs, RNNs and transformers to achieve state of the art results. More recent studies are now pushing the boundries by applying transfer learning to improve performance of sentiment analysis in specialized cases. A domain adaptation method was proposed by Peng et al. (2018) that used a shared representation to transfer knowledge across domains and improve the performance of sentiment analysis. Some of the most recent trends follow Aspect-based sentiment analysis (ABSA) that aim to identify sentiment towards specific aspects or attributes of a product or service.

Where traditional text classifiers have always lacked is in learning underlying context in the text. That is where NNs, word embeddings and transformers have come into play in the evolution of sentiment analysis. The primary reason for the limitations of traditional ML classifying methods lies in fact that they are built upon word representations and generalize on training data made available in the form word vectors. Hence, the next generation of modelling methods fixes this by utilizing similarities between words as a distance or angle to capture deeper relationships between words and capture contextual relevance Maas et al. (2011).

In this project I have established the limitations of traditional ML algorithms to plateau in their ability to generalize and classify a sentiment label based on word representations. After I establish this, I introduce the capture of contextual relationship to improve the performance of the classification task by making use of word embeddings and deep learning methods to learn temporal dependencies in user posts. One of the drawbacks of working with social media user posts is that the posts are small in length and often written as short form opinions of users without proper motivations and explanation Wang et al. (2018). This creates a challenge in limiting the capability to learn contextual associations. To address this, I make use of pre-trained word embeddings to make up for the limitations of learning from the given dataset.

2 Related Literature

2.1 Word Embeddings

To successfully accomplish a sentiment classification task, we need to understand and capture the semantic compositionality of the data Sag et al. (2002). Some of the most popular advances in this field of NLP have been made using word embeddings. They are a type of distributed representation of words in a vector space that capture semantic and syntactic similarities between words. Word embeddings derive the relationship between words by using the cosine distance between their vector representations. Similar words have their vector representations also closer to each other in terms of cosine distance. In the context of a sentiment analysis, Word Embeddings can capture semantic and syntactic features of words like word associations and context. Another benefit of Word Embeddings was discussed by Bojanowski et al. (2017) where the author proposed a skip-gram model approach to learn subword information that would help in handling out of vocabulary words by capturing both the meaning of individual subwords and the meaning of the full words.

Therefore embeddings for phrases can also be built by adding the individual vectors of the words in the phrase. Two of the most popular algorithms for obtaining word embeddings are Word2Vec and GloVe. Both the models are different in how they are trained. Word2vec is FFN which is either modelled on the skip-gram model or the continuous bag-of-words or CBOW model. GloVe is matrix factorization technique which constructs large matrices of co-occurrence (words x context) which is then factorized to a lower dimension matrix (words x features) in which each row in the matrix gives the vector representation of the corresponding word. Pennington et al. (2014) have established GloVe to be the superior model for the unsupervised learning of word representations which outperforms other models like word2vec in word analogy, similarity and NER tasks. FastText is an extension of Word2Vec that was introduced by Facebook AI Research in 2016 that can generate embeddings for sub-word units such as character n-grams. FastText was shown to outperform Word2Vec using subword information to improve the quality of word embeddings in the original FastText paper. FastText represents each word as bag of character n-grams where each character has a vector representation Bojanowski et al. (2017). ELMo (Embeddings from Language Models) is another deep contextualized word embedding algorithm that has been shown to provide sate of the art results for a wide variety of NLP tasks. Jain et al. (2021) made a comparative study of Word2Vec, GloVe, FastText and ELMo by testing them for a hate speech detection task and reported that ELMo gave the best results followed by Fast-Text and GloVe.

2.2 Deep Learning in NLP

With the computational resources available today, Deep Learning has become a goto tool for complex ML tasks such as NLP, image analysis and speech recognition. Deep Learning enables a user to deploy multiple layers to learn complex features and transformations. In the context of NLP this means that NNs can enable learning of syntactic features fromthe given corpus. There is a plethora of research that shows how NNs have been applied for NLP tasks and have shown great potential as discussed by Zhang et al. (2018).

Recurrent NNs came into the field of NLP to overcome the barrier of learning temporal dependencies between words that give contextual meaning to the sentence. RNNs are a category of NNs where the neurons form a directed cycle. RNNs are able to process a sequence of inputs with each output being dependent on previous computations. This is what makes RNNs very popular in addressing NLP tasks. This was never possible by using the traditional ML classifying techniques which rely on generalizing through training on word vectors. The calculations conducted by a RNN were summarized by Wang et al. (2016) as follows:

1. The input at time step t, is x_t, which can be a one hot vector or word embedding for a NLP task

2. h_t is the hidden state which is calculated using the input at the current time step and the previous hidden state, given by:

$$h_t = \sigma(Ux_t + Wh_{t-1})$$

3. The output, o_t, at time step t for a sentiment classification task would be a vector of probabilities across all the sentiment categories, given by:

$$o_t = softmax(Vh_t)$$

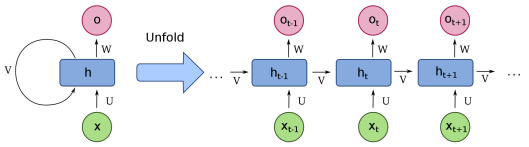


Figure 1: Unfolded RNN

RNNs typically suffer from the limitation of the vanishing gradient problem and the exploding gradient problem. In an RNN, the same set of weights is used for each time step, which means that the gradient is computed recursively by multiplying the current gradient with the derivative of the activation function and the weight matrix at each time step. However, this multiplication can result in the gradient becoming exponentially small as it propagates backwards through time, particularly when the weight matrix is initialized to small values or when the activation function saturates at large or small input values. As a result of the vanishing gradient problem, RNNs may have difficulty capturing long-term dependencies between sequential inputs, particularly when the dependencies are spread out over many time steps. This limitation can be particularly problematic for tasks such as natural language processing, where the meaning of a sentence can depend on the context of the entire document. The exploding gradient problem can occur when the weight matrix is initialized to large values or when the gradient is amplified by the derivative of the activation function. When the gradients become too large, they can cause the weights to update too quickly, leading to unstable training and poor model performance. To address the exploding gradient problem, researchers have proposed various techniques such as gradient clipping, weight regularization, and adaptive learning rate methods, which can help stabilize training and prevent the gradients from becoming too large. To address some of these limitations, researchers have proposed various modifications to RNNs, such as LSTMs and GRUs, that are better able to handle long-term dependencies and vanishing gradients. Additionally, newer models, such as transformer-based models, have shown better performance on many NLP tasks, including sentiment analysis.

One of the most popular RNN variant is the LSTM (long-short term memory). The basic building block of an LSTM is called a cell, which is composed of several key components including an input gate, a forget gate, a memory cell, and an output gate. The input gate controls how much of the new input should be stored in the memory cell, while the forget gate controls how much of the previous memory cell state should be retained. The memory cell is responsible for storing the information over time, and the output gate controls how much of the memory cell state should be used to compute the output. LSTMs use a combination of gating mechanisms and memory cells to selectively store and update information over time, allowing them to capture long-term dependencies in sequential data.They have been proven to provide state of the art performance for several language modelling tasks by allowing for connections between units allows them to use their internal states to process sequences of inputs and capture temporal dependencies. This makes them particularly well-suited for tasks such as natural language processing, speech recognition, and time series analysis. The calculations conducted by a LSTM were also summarized by Wang et al. (2016), for an input x_t with previous hidden state h_{t-1} and the previous memory state c_{t-1}. The hidden state and memory is updated with the following equations:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$f_t = \sigma(W_f x_t) + U_f h_{t-1} + b_f$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o$$
$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_i \odot g_t$$
$$h_t = o_t \odot tanh(c_t)$$

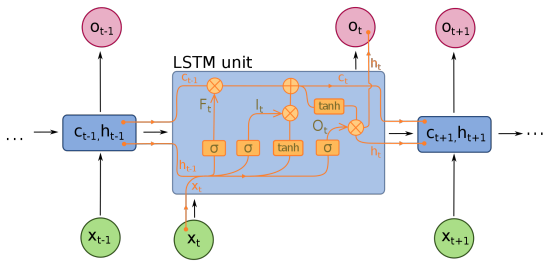


Figure 2: LSTM Unit

where $\sigma(.)$ denotes the logistic sigmoid function. The operand $\odot$ denotes element wise vector product. For each time step t, i_t denotes the input gate, f_t denotes the forget gate, o_t denotes the output gate, c_t denotes the memory cell and h_t is the hidden unit. W, U and b are the parameters ofthe LSTM.

Another popular variant of RNNs is GRU or Gated Recurrent Unit which is similar to a LSTM but without the output gate Chowdhury and Zamparelli (2018). While LSTMs have three gates (input, forget, and output), GRUs have two gates (reset and update). GRU was introduced by Cho et al. (2014) and it aims to solve the vanishing gradient problem that occurs with standard RNNs. Because of the lacking output gate, the memory cell contents are copied into the network at each time step. GRUs are supposed to be simpler and faster than LSTM and generally produce par results if not better than LSTMs. They have been shown to perform well on a wide range of tasks, including machine translation, speech recognition, and sentiment analysis. As before, calculations conducted by a GRU were also summarized by Wang et al. (2016) where the parameters update as per the following equations:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$
$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$
$$\hat{h}_t = tanh(W x_t + U(r_t \odot h_{t-1}))$$
$$h_t = (1 - z_t)h_{t-1} + z_t \hat{h}_t$$

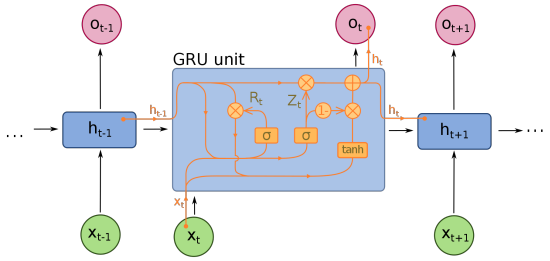


Figure 3: GRU Unit

where $\sigma(.)$ denotes the logistic sigmoid function. The operand $\odot$ denotes element wise vector product, r_t is the reset gate, z_t is the update gate and $\hat{h}_t$ denotes the candidate hidden layer.

Bi-directional language models (Bi-LMs) are useful for sentiment analysis because they can provide contextualized embeddings of words, which capture the meaning of a word in a particular context. This is important for sentiment analysis, as the meaning of a word can vary depending on the context in which it is used. One common approach to implementing Bi-LMs is to use a bi-directional RNN (Bi-RNN). In a Bi-RNN, the input sequence is processed by two separate RNNs, one processing the sequence in forward order and one processing it in backward order. The hidden states of the two RNNs are then concatenated to produce a final representation of each word in the sequence that takes into account both the preceding and succeeding words. Contextual word representations are derived from Bi-directional Language models (Bi-LMs) which combine a forward and backward language model to jointly maximize the likelihood in both directions as shown by Peters et al. (2018):

$$\sum_{k=1}^{n} (logP(t_k|t_1, ..., t_{k-1}; \overrightarrow{\Theta}) + logP(t_k|t_1, ..., t_{k-1}; \overleftarrow{\Theta}))$$

Overall, multiple studies have shown that RNNs variants can perform comparatively better or worse depending upon the specific task at hand. For the purpose opf the sentiment analysis that is being attempted here, experiments can be conducted to compare word embeddings with these different architectures.

3 Data

The data used in this projectis a compilation of reddit comments made publicly available on Kaggle. The data set contains the posts and comments on Reddit which mention the terms “climate” and “change”. The data is quite an extensive corpus of comments with comments scraped till 2022-09-01 and contains approximately 40 million comments. The dataset has been limited to include data about subreddit, NSFW(Not Safe For Work) tag, comment date, comment text, link to the comment, the score of the comment (upvotes/downvotes) and an analyzed sentiment for the comment. The dataset was chosen because it had a sentiment score which could be converted into classes. This helped me in the evaluation of the models. It was not clarified by the author as to what methodology had been used in calculating the sentiment scores. But a sample check was sufficient to infer that the sentiment score reflected the true sentiment label of the comments. In the interest of computational resources available to me, I have chosen to use only 1.25% of the dataset which amounts to a corpus of more than 50k comments.

There was a standardized preprocessing that was applied to the data. I made use of the large english library in SpaCy to remove stop words and punctuations. Since here I was dealing with social media data, I also removed any url links. An exception was made to the stop words library by removing

'no' and 'not' from the standard SpaCy stop words list. This was done keeping in mind that the task at hand is a sentiment analysis where these 2 words can be important in the context of the sentiment identification.

4 Method

4.1 ML Algorithms

Ignoring any underlying contextual relationships within sentences and phrases, any sentiment analysis is at its core, simply a text classification problem. In our case, because the data is labelled, this can be modelled as a supervised text classification problem with the intention to showcase the limits of using traditional classification algorithms. For this effect we setup and train the following ML classifiers to identify a baseline as well as to estimate what is the best classifier we can train to give us the best predictions.

Classifier	Params
Naive Bayes	default
Logistic Regression	max_iter=1000
XGBoost	100 estimators, lr = 0.1, depth = 10

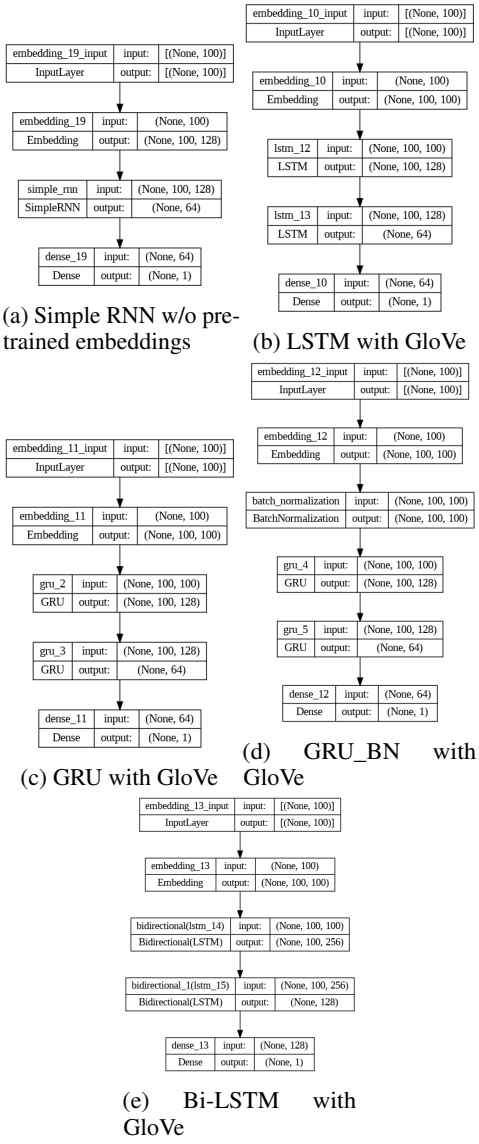
The above classifiers help in establishing a baseline F1-score of 71% with the Naïve bayes. But inspecting the other classifiers we realise that the we hit a ceiling of F1-score at 74% for xGBoost classifier and we can achieve F1 score of 80% with Logistic Regression.

4.2 RNNs

After establishing this baseline and plateau performance, we then move to the main idea of this project i.e. employing word embeddings for finding semantic relations in the text and finding a model that best classifies the sentiment of the user opinions. To this effect we will use the following RNN variants:

RNN Variant	Pre-trained Embeddings
Simple LSTM	None
LSTM	Glove-twitter-27B.100d
GRU	Glove-twitter-27B.100d
GRU w/ Batch Normalization	Glove-twitter-27B.100d
Bi-LSTM	Glove-twitter-27B.100d

The architecture of each of these variants is given in the figures. We first use a simple RNN architecture without any pre-trained embeddings, a keras tokenizer with a vocabulary of 5000 words and one dense layer. We follow this up by constructing a Deep LSTM model and another comparative model



(a) Simple RNN w/o pre-trained embeddings

(b) LSTM with GloVe

(c) GRU with GloVe

(d) GRU_BN with GloVe

(e) Bi-LSTM with GloVe

Figure 4: RNN Architecture for chosen Models

with GRU. In both these models we use pre-trained embeddings from GloVe vectors. For this project, I tried with 2 different embeddings, first the 6Bn tokens trained on Wikipedia2014 + Gigaword5 and secondly, the vectors from 27Bn token trained on 2 Bn tweets. I found the results to be slightly more favorable with the Twitter based embeddings which makes sense since my dataset is also from social media. This pretrained embeddings were used to construct an embedding matrix for the vocabulary. I use this matrix as weights for the embedding layer in the LSTM and the GRU models. The other aspects of these experiments were the introduction of 20% dropout. Dropout to counteract overfitting and to enable the NN to learn robust features and reduces the dependency on any particular feature. Unlike regular dropout, where the same inputs are

randomly set to zero for all time steps, recurrent dropout only sets to zero the inputs that are fed into the hidden state of the RNN at a given time step. This is because the hidden state of an RNN is fed back into the network at the next time step, making it important to keep a consistent set of inputs for each time step. But the recurrent dropout comes with a very high computational cost and was found to greatly increase the training time.

5 Results

5.1 Evaluation Criteria

F1 score is a good metric in sentiment analysis because it takes into account both precision and recall. Precision measures the proportion of true positive predictions out of all positive predictions, while recall measures the proportion of true positive predictions out of all actual positive instances. In sentiment analysis, we want to correctly classify both positive and negative sentiments, so both precision and recall are important. F1 score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance on both positive and negative classes.

Classification Results	f1 Score
Naive Bayes	0.71
Logistic Regression	0.80
XGBoost	0.74

We can infer from the results of the ML classification algorithms that the best performance can be obtained from training a logistic regression classifier which yields a F1 score of 82%. xGBoost don't perform as well as the Logistic Regression classifier. We move now to the contextual language modelling based RNNs.

RNN Variant	f1 Score
Simple RNN	0.72
Deep LSTM	0.75
GRU	0.74
GRU w/ BN	0.77
Bi-LSTM	0.70

The impact of pre-trained embeddings can be seen from the results as the results of the Simple RNN with keras tokenizer gives inferior results to the models utilizing GloVe embeddings. The best performance within the RNN variant implementations wih GloVe vectors was obtained with the GRU with batch normalization with an overall f1 score of 77%.

6 Discussion

In this project I have been able to show that to tackle a sentiment classification problem we cannot rely alone on word vector based ML classifiers. It will result in a plateaued performance and we will not be able to improve the performance of our classifier beyond a certain point. The importance of contextual language modelling is established by the use of RNNs and word embeddings. This was expected because learning word embeddings would require a huge training corpus and very steep computational resources. That is why we make use of pretrained GloVe vectors which have been trained on twitter tweets.

Our best performing RNN model was the Gated Recurrent Units model which utilizes Bathc Normalization. The benefits of GRUs are manifold, the biggest being the speed of training due to its ability to deal with the vanishing gradient problem. The overall results from the trained RNN variants maybe reported to be less than what we saw for the logistic regression classifier, but the important point to note here is that for the logisitic regression classifier, the performance cannot be improved further. A side experiment was conducted with cross validation to fine tune the logisitic regression classifier which yielded no improvement over the reported f1 score. The same is not true for the RNN implementations. It is my belief that with better hyperparameter tuning to suit the dataset upon which this study is modelled, we can easily improve the quality of the results. For instance, one way to improve the results would be to use recurrent dropout. This greatly impacts training time but was showing promising improvements in the experiments conducted.

7 Limitations

This study was pretty exhaustive in terms of the classifiers employed and NN architectures explored for fulfilling the task of sentiment analysis. While, only the best performing classifiers were reported in the previous sections, during the experiments a wide plethora of classifiers were trained and tested. One of the key observations was that the gradient based classifiers were performing better than the ensemble classifiers. While designing the experiments, it was the intiutive expectation that RNN

variants would vastly outperform the ML classifers, but in actuality the the improvement was not as ground breaking. The follow up was a thorough experimentation with various word embeddings and even a transformer model. These results were not reported with the main findings of this project because of unfavourable results. The lack of sharp improvement in the quality of results led us to question the quality of the data being used. While the experiments were being designed and tess with small amounts of data to compromise on computational resources available, the final experiments used a significant amount of training data but still did not yield significant improvement. Revisting the reddit posts in the data, I as able to observe that there are a lot of texts and posts that might be more suitably classified as NEUTRAL. This might be a reason for why we see the quality of the results taper off. This leads to a hypothesis that an alternate way of looking at this problem is as a multi-sentiment classification problem rather than a binary-sentiment classification problem.

Another interesting observation about the data is that in the experiments I have ignored the additional information about the sub-reddits where these posts and comments are made. The Reddit community is known to be notoriously polarizing full of trolls and strong opinions with little to no filters. This leads to some sub-reddits being more 'toxic' than others. This can lead to a certain bias in the sentiment of posts and comments within such subreddits. The experiments designed in this project do not tackle this complexity.

Another limitation that was discovered and then abruptly corrected was that the raw texts not being pre-processed for stop words and punctuations was leading to lower quality results with RNNs. This was even more pronounced in the intial simple LSTM NN which does not employ pre-trained word embeddings. The quality of that output was much lower than the ML classifier results and had to be corrected with appropriate pre-processing.

Another potential improvement that could have been tried in this study was the use of an alternative to GloVe embeddings. As was reported earlier in this report, some researchers have reported ELMo to show better results than GloVe embeddings for sentiment analysis tasks. Depending upon the data being analyzed, even FastText has shown better results than GloVe.

The last limitation that I would like to talk about is the intensive computational resource dependency for this problem. As was shared earlier in the report, the dataset is actually quite extensive with nearly 40 million reddit comments. Tackling the entire dataset is a very very computationally expensive task and hence, for the purpose of this project, I had to work with a fraction of the overall data.

8 References

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Shammur Absar Chowdhury and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Mary Harper. 2014. Learning from 26 languages: Program management and science in the babel program. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, page 1, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Minni Jain, Puneet K. Goel, Puneet Singla, and Rahul Tehlan. 2021. Comparison of various word embeddings for hate-speech detection.

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.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135.

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, Melbourne, Australia. Association for Computational Linguistics.

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.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Conference on Intelligent Text Processing and Computational Linguistics*.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. 2018. An LSTM approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*, pages 214–223, Hsinchu, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2428–2437, Osaka, Japan. The COLING 2016 Organizing Committee.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.