|  |
| --- |
| **Improving Upon REGMAPR** |

|  |  |
| --- | --- |
| **Dylan Sholtes**  Computer Science Department  Southern Illinois University Edwardsville  *dysholt@siue.edu* | **Christie McDaniel**  Computer Science Department  Southern Illinois University Edwardsville  *chmcdan@siue.edu* |
| **Tyler Norton**  Computer Science Department  Southern Illinois University Edwardsville  *tnorton@siue.edu* | **Ruslan Cahnovsky**  Computer Science Department  Southern Illinois University Edwardsville  *rcahnov@siue.edu* |

**Abstract**

Textual entailment is a notable NLP task that remains an active area for research and can be useful in information extraction and retrieval among other applications. Due to requiring the understanding of semantics between sentences, it remains one of the more complex NLP tasks. Many previous models addressing this problem required complex implementations. We propose to improve upon the REGMAPR model put forth by Brahma in 2018 which addresses this problem and focuses on simplicity and efficiency through the use of a Siamese architecture. This architecture is then enhanced with matching and paraphrase features and four types of regularization are used to further optimize results. We first ran our baseline model on the SICK dataset and collected the Pearson Correlation values for training and testing and achieved results on par with that of the original model. We then tested the model by removing just the matching feature, removing just the paraphrase feature, and then removing both and saw our values drop. Following this we returned to using REGMAPR and adjusted hyperparameters including how many epochs the model was tested over and achieved better results than the baseline model. We also tested the model using different optimizers including Adamax, ASGD, and Adadelta.

**1. Introduction**

While working with languages and sentences can come intuitively for humans, machines must be taught to understand the intricacies and relationships within language and between sentences. In particular, determining contradictions between claims can be natural for humans, but for machines there still lies room for improvement. Natural Language Processing (NLP) is still growing and has not yet caught up to the abilities of humans. Improving upon machines’ ability to identify contradictions within claims would allow for advances within fact-checking, the identification of fake news, analysis of text, and many other areas. Therefore our goal is to explore a novel approach to NLP and more specifically Natural language Inference (NLI) to improve upon machines’ ability to identify contradictions within pairs of sentences.

The input to our model will be pairs of sentences, which will be drawn from the Sentences Involving Compositional Knowledge (SICK) dataset, and the training data will also be accompanied by labels of either entailment, contradiction, or neutral as well as a relatedness score [9]. We will use the REGMAPR architecture to output a judgment on how related the two sentences are. Sentences are scored on a scale of 1-5 based on how related they are, sentences with little to no relationship with each other are in the 1-2 range, while closely related sentences are in the 4-5 range.  Due to limitations, we have focused on using only the relatedness score label and then from that confirming what kind of relationship it is. We have also simplified the range in the Classification layer from 1-5 to 0-1.The figure below shows our intended work-flow going forward.

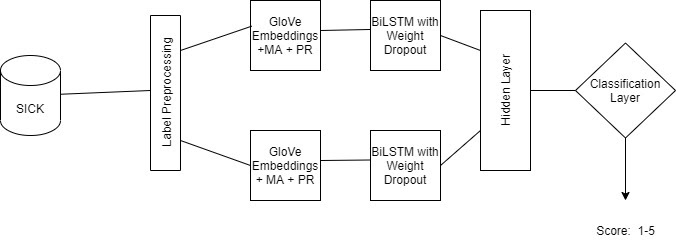


Figure 1: Shows an example of data preprocessing to training and making classifications.

**2. Related Work**

Several approaches exist for tackling NLP problems, among which REGMAPR takes on a more novel approach, first being used in 2018 [1]. REGMAPR varies from most established approaches by focusing on simplicity, such as how it applies a Siamese architecture. For established approaches, REGMAPR is most similar to the works of Wang and Jiang who use a compare-aggregate model to match sequences from sentence pairs [2]. REGMAPR shares similarity in approach since Wang and Jiang’s method relies on vectors and comparisons, which REGMAPR does as well though it uses a different structure. Whereas REGMAPR uses the Siamese architecture, Wang and Jiang’s use the compare-aggregate framework. This framework focuses on comparing vector representations of smaller units such as words, “and then aggregate these comparison results to make the final decision” [2]. He and Lin used a similar approach, also modeling pairwise word interactions and using a similarity focus layer [3]. Parikh et al. (2016) also used comparisons with words and attention-weighted versions to produce comparison vectors for use in aggregation [5]. All of the aforementioned models take an approach where they split up the data even further and apply vectors and comparisons to gain their results and high accuracy. Weaknesses of these models can be related to the comparison function used, since enough attention is not always paid to it. Additionally, the extent to which these approaches can be applied has not fully been explored since they haven’t been widely tested [2].

Differing approaches include NLP models that use a BiLSTM with intersentence attention. Examples include a proposed model by Liu et al. (2016), which uses BiLSTM with a sentence encoding module to extract relationships between sentences [4]. Other sentence encoding-based models include the, “LSTMs-based model, GRUs-based model, TBCNN-based model and SPINN-based model” [4]. However, these suffered from not fully utilizing contextual information, which Liu et al. improved upon in their model. Similarly to REGMAPR, however, is that Liu et al.’s model employs a Siamese architecture where it has two identical sentence encoders. Mueller and Thyagarajan (2016) made use of the aforementioned LSTM-based model, adapting it for use in a Siamese architecture [10]. In their results, they showed how a simple LSTM model could be trained with enough data to, “learn a highly structured space of sentence-representations that captures rich semantics,” while sidestepping complexity of prior LSTM and other models [10].

From the prior work and models, it would seem that approaches that highlight simplicity are currently gaining traction and proving to be just as successful, if not more so, than more complex models. In particular, this is seen in the increased use of the Siamese architecture being applied to NLP problems where it had not been before. Utilizing a Siamese architecture is becoming more state-of-the-art as it proves itself to be more effective. This can even be seen in REGMAPR’s adaptation of the architecture which allowed it to remain much simpler in implementation while producing highly accurate results as seen in those published in the SNLI Corpus [6]. While the task of classifying sentences can be performed by hand and was performed this way for the creation of the datasets, for vast amounts of data or complex sentence pairs, models such as the ones mentioned prove to be very efficient and thus can save time from a sometimes menial task. Additionally, advances in automating the task of sentence classification could allow for more instant analysis of text which could prove beneficial in many different fields.

**3. Materials and Method**

**3.1 Dataset and Features**

The primary dataset we will use for training and testing is the Sentences Involving Compositional Knowledge (SICK) dataset [9]. This dataset contains nearly 10,000 pairs of sentences labeled for entailment relation as well as annotated for relatedness in meaning shown as a gold score on a 5-point rating scale. The sentence pairs were manually labeled through human ratings collected via a crowdsourcing study which made use of the Amazon Mechanical Turk marketplace. The breakdown of the dataset’s relatedness and entailment is pictured in Figure 2. This breakdown shows a large skew in the dataset, with many of the relatedness scores being in the 3-5 range. For textual entailment tasks, only 14% of the sentence pairs are contradictions, while 57% of the sentences are neutral, and 29% of the sentences are entailments.

|  |  |  |  |
| --- | --- | --- | --- |
| Relatedness | | Entailment | |
| [1-2) range | 923 (10%) | NEUTRAL | 5595 (57%) |
| [2-3) range | 1373 (14%) | CONTRADICTION | 1424 (14%) |
| [3-4) range | 3872 (39%) | ENTAILMENT | 2821 (29%) |
| [4-5] range | 3672 (37%) |  | |

Figure 2: Distribution of gold scores in the SICK dataset [9].

The Paraphrase Detection Database version 2.0 (PPDB) made by Pavlick et al. (2015) will also be utilized within the paraphrase detection layer of the REGMAPR model [7]. PPDB contains over 100 million paraphrases in 16 different languages. Additionally, the second version of this dataset contains improvements in many areas including scoring for ranking paraphrases as well as entailment relation in each pair (seen in the symbols in Figure 3).

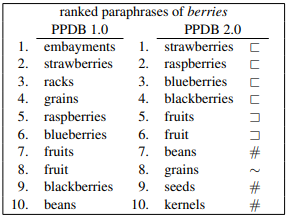


Figure 3: Examples of paraphrases from PPDB version 2.0, including improvements between versions.

For testing the baseline model with the SICK dataset, data was preprocessed by converting the semantic relatedness score from a scale of 1 to 5 down to a scale of 0 to 1 for the classification layer. This is done through a simple python script that uses pandas to read the dataset and convert the scale down, and returns the dataset as a tsv. The script also removes other columns present in the SICK dataset, leaving only the two sentences and the target column. REGMAPR makes use of two features, a matching (MA) feature, as well as a paraphrase detection (PR) feature. The matching feature detects an exact word match between the pair of sentences, while the paraphrase detection feature makes use of the PPDB to detect paraphrases between the sentence pairs. These features add an additional two layers to the word embeddings. REGMAPR makes use of GloVe for word embeddings, with an embedding size of 300, giving a total of 302 layers with the embeddings and the features.

**3.2 Methods**

REGMAPR approaches NLP with a Siamese architecture, which makes two or more identical subnetworks that share weights and can compare feature vectors. The input sentences are then encoded, “using a BiLSTM and the representations are composed by computing the element-wise absolute difference and product” [1]. Further, the embeddings of words are augmented with a matching feature to determine if the same words are shared between sentence pairs. REGMAPR utilizes an external database of semantically related words to identify semantic dependence without being too constricted by the aforementioned embedding; this adds a paraphrase feature to the embedding of words, resulting in two additional dimensions for the embeddings that capture syntactic and semantic interaction between the sentence pairs. REGMAPR is defined by its base Siamese architecture and two dimensions of embeddings on words. However, to avoid overfitting regularization is applied; for REGMAPR specifically, four types of regularization were used to train the models and then compare the results.

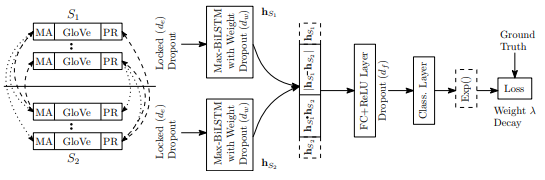


Figure 4: Schematic diagram of REGMAPR model from Brahma (2018) [1].

More specifically, REGMAPR works by encoding each sentence into a vector using BiLSTM, then max-pooling is used on the intermediate states produced by BiLSTM. Feature vectors are then made via the encodings of the sentences which are, “composed by concatenating the element-wise absolute difference and element-wise product with the original vectors” [1]. The feature vector is passed through a fully connected layer and then ReLU activation, and finally a classification or scoring layer.

REGMAPR also makes use of four types of regularization. Locked Dropout is applied after the embedding layer, Simple Dropout is applied after the ReLU activation, Weight Dropout is applied to recurring weights in the BiLSTM encoder, and L2 Weight Decay regularization is applied for the loss.

Combining the matching (MA) and paraphrase (PR) features results in an equation shown in Figure 5. It uses a 300 dimensional GloVe, where GloVe is an unsupervised learning algorithm to obtain vector representations of words where *t* represents a word in the sentence [8]. The set of words are denoted by T(S3-i) where i ∈ {1, 2}, with the first sentence being represented by S1, and the second sentence being represented by S2. The purpose of using T(S3-i) is for when computing ESi(*t*), the word *t* is checked against the other sentence in the pair.  𝟙 represents the indicator function, stating whether the word has an exact match in the set of words T(S3-i) in the matching feature, or if T(S3-i) contains a paraphrase match in the set P(*t*) in the paraphrase feature. P(*t*) represents the set where P(*t*) = {*t’* | *t’* is a paraphrase in the PPDB}[1]. Both the matching and paraphrase features are binary features, giving results of 0 or 1.

Figure 5: Equation for the combination of matching and paraphrase features.

For our hyperparameters, we used an embedding dimension size of 300 and optimized using Adam with a learning rate of 0.001. For regularization, hyperparameters of 0.4 for Locked Dropout, 0.4 for Simple Dropout, and 0.2 for the Recurrent Weight Dropout. The model is trained over 10 epochs, with batch sizes of 64 for the training set, and batch sizes of 128 for the validation set. These hyperparameters were chosen as they were used by Brahma (2018) who originally proposed the REGMAPR model and achieved good results [1]. This gives our group a good baseline for comparison against Brahma’s model and allows us to tune these hyperparameters to increase the accuracy.

**4. Results**

The following is a table of our results achieved where all trials were run using a training set of size 7,000 and a test set of size 1,500. Pearson correlation values are also shown for both training and testing of each model. Trials 1-4 made use of the baseline REGMAPR model made by Ben Cunningham [11]. Trials 5-15 made use of the full REGMAPR model with additional changes as noted. The best training and test performances were achieved in trial 14 with Pearson values of 0.9516 and 0.8556 achieved for training and test performance. Next highest results were achieved in trial 13 with Pearson values of 0.9499 and 0.8534 achieved for training and test performance. This would suggest that training over 10 epochs and using a LSTM hidden dimension size of 300 proved most effective. Changing the dimension of the hidden layer in the model in addition to the aforementioned changes increases performance further to achieve the best performance found in our trials.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Pearson Correlation  (r) Training | Pearson Correlation  (r) Test | Training Size | Test Size |
| 1 | BASE + REG + MA + PR (REGMAPR) | 0.7729 | 0.6825 | 7000 | 1500 |
| 2 | BASE + REG + MA | 0.7347 | 0.6715 | 7000 | 1500 |
| 3 | BASE + REG + PR | 0.7529 | 0.6602 | 7000 | 1500 |
| 4 | BASE + REG | 0.76703 | 0.6416 | 7000 | 1500 |
| 5 | REGMAPR lr=0.005 | 0.8346 | 0.7196 | 7000 | 1500 |
| 6 | REGMAPR 100 epochs | 0.943 | 0.784 | 7000 | 1500 |
| 7 | REGMAPR lr=0.005 100 epochs | 0.9336 | 0.8017 | 7000 | 1500 |
| 8 | REGMAPR Adamax 100 epochs | 0.8608 | 0.7398 | 7000 | 1500 |
| 9 | REGMAPR ASGD 100 epochs lr=0.01 | 0.1924 | 0.2474 | 7000 | 1500 |
| 10 | REGMAPR Adadelta 100 epochs lr=1 | 0.8661 | 0.749 | 7000 | 1500 |
| 11 | REGMAPR Recurrent Dropout =0.4 | 0.9132 | 0.7379 | 7000 | 1500 |
| 12 | REGMAPR embed dim=600 hidden dim = 128 | 0.8973 | 0.771 | 7000 | 1500 |
| 13 | REGMAPR 10 epochs lstm-hidden-dim = 300 | 0.9499 | 0.8534 | 7000 | 1500 |
| 14 | REGMAPR 10 epochs  lstm-hidden-dim 300 hidden dim = 300 | 0.9516 | 0.8556 | 7000 | 1500 |
| 15 | REGMAPR Stacked BiLSTMS | 0.9335 | 0.7986 | 7000 | 1500 |

Table 1: Table of the results of running various trials on the REGMAPR model.

The following chart gives a visual representation of the results for all 15 trials. It compares the Pearson correlation values for training and test within each trial, as well as puts each individual trial’s results in perspective with every other trial completed. Pearson correlation values for training are pictured in blue (to the left in each trial), and Pearson correlation values for testing are pictured in red (to the right in each trial).

Figure 6: Graphical comparison of Pearson correlation values for training and tests run using REGMAPR.

**5. Discussion**

**5.1 Baseline Model Testing**

The baseline model we started from was a full implementation of REGMAPR, with both matching (MA) and paraphrase (PR) features implemented [11]. Removing both of these features had a significant decrease in Pearson correlation values. After running tests where either the paraphrase or matching feature was removed, we concluded the matching feature had a larger impact on the Pearson correlation values than the paraphrase feature did. However, using both paraphrase and matching features together yielded the best results for the baseline model. The baseline model was a few years old, and thus contained many deprecated PyTorch features. Before running any tests, we fixed these deprecation issues and saw improvements to the loss when training models.  Increasing the number of epochs the model was trained on from 10 to 100 brought large improvements to both training and test Pearson correlation values. Values were further improved by increasing the learning rate from 0.001 to 0.005.

**5.2 Further Testing**

From this point, we decided to experiment with different torch optimizers and criterion. Many of these optimizers gave good results, with the exception of asynchronous stochastic gradient descent, which severely reduced the Pearson correlation values for both training and test sets. This is most likely due to our labels being a relatedness scale, which stochastic gradient descent optimizers perform poorly on, as they have trouble escaping local minima. We next tried using a cross entropy loss function instead of mean squared error; however, since there is only a single class, the relatedness score, cross entropy loss did not function with our model, and was not included in our results. Based on our results, we concluded that for sentence relatedness tasks, mean squared error was the best criterion for our model, and Adam was the best optimizer. These are the same criterion and optimizers that Brahma used when proposing the model [1]. After trying different optimizers and criterion, we changed the hyperparameters for the dropout rates of the three types of dropout used. We changed the values of these hyperparameters in the range of 0.1 to 0.4, and none of these had any significant impact on Pearson correlation values, such as the recurrent dropout of 0.4 as shown in Table 1. We also attempted to increase the accuracy by using two identical stacked BiLSTMs instead of a single BiLSTM. This led to higher overfitting and a worse Pearson correlation coefficient when compared to a single BiLSTM with a hidden size of 300. This could possibly be due to the large number of hidden features present in the BiLSTM, leading to larger amounts of overfitting when stacking them. Reducing the number of hidden features present when stacking BiLSTMs could possibly reduce this. While running the tests, there was some overfitting present, with many of the Pearson correlation values for the training set being over 0.1 higher than the Pearson correlation values on the test set. We believe this is due to the imbalance in relatedness scores for the SICK dataset. In the SICK dataset as pictured in Figure 2, 3,872 pairs of sentences are within the 3-4 range, and 3,672 pairs of sentences are within the 4-5 range, compared to only 923 pairs of sentences within the 1-2 range. This would lead to many of the predictions being within the 3-5 range.

**5.3 Limitations**

Unfortunately, one of our limitations was that we were unable to transform the SNLI dataset for use with our model, which would have allowed us to confirm if the overfitting was due to the dataset or the model, or potentially both. Potential overfitting due to training was mitigated by using four types of regularization, as well as using a small learning rate of 0.005 with the ADAM optimizer. Using higher learning rates with ADAM led to instability when reaching convergence due to ADAM maintaining a rolling geometric mean of recent gradients. The three types of dropout used in the REGMAPR model helps prevent overfitting by making the training noisier and preventing the model from attempting to fix mistakes from previous layers by having network layers coadapt. Tests run with 100 epochs had a higher amount of overfitting than tests run with 10 epochs; however, the model had higher Pearson correlation values on the test set when trained over 100 epochs, with the exception of tests ran with an LSTM hidden size of 300. The number of epochs the model is trained over can be reduced to between 10 and 100 to control the amount of overfitting present when training the model.

The other limitation as we stated above is how heavily skewed the SICK dataset is. Our assumption is that the overfitting was most likely due to the dataset and getting a more fleshed-out dataset would prevent a lot of the overfitting. Having only about 10,000 sentences, with the distribution being heavily layered on the 3-5 relatedness score side, makes it very hard for the training to be as accurate as we would have liked. With the SNLI dataset, the amount of data we would be able to use would have increased by an enormous amount. The SNLI dataset contains around 570,000 sentences all ranging between 1-5 relatedness score. We could potentially use around 80% of the dataset for training and the other 20% for testing. This would make it so 456,000 sentences are used for training and then the remaining 114,000 would be used for testing. Given that the difference in training datasets would be over 400k, it would be safe to assume the overfitting would most likely not happen with the SNLI dataset, of course this is only if the overfitting is due to our dataset and not the model. Another limitation would be that we were not able to test some runs with 100 epochs. This was because of the amount of time it would take to run those tests. Usually, it would take around half an hour to complete a run with 10 epochs, but when we changed it to 100 epochs it increased that time to several hours or more. Trying to complete five extra runs with 100 epochs and then also some different parameters to differentiate those runs from each other would be very time consuming and the results might not be significantly different from each other to warrant the time commitment.

One other limitation is that we are using the Pearson correlation values for checking our accuracy. For sentence relatedness tasks, accuracy is not a valid measure, and other measures must be used. These methods can include Pearson correlation values, Spearman correlation values, and other measures such as cosine distance. With the SNLI dataset we would be able to actually see the accuracy of our training and testing. Instead, we are only seeing correlation values between the relatedness score. An additional limitation would be that the paraphrase detection feature we implemented does not seem to make proper use of the PPDB. It checks the dataset and then finds any words that are paraphrased from the current word. With our paraphrasing detection feature not properly implementing PPBD, it could be one of the reasons why we could not fully incorporate the SNLI dataset.

**6. Conclusion and Future Work**

We started out with a baseline model with both matching (MA) and paraphrase (PR) features implemented. From our baseline, we fixed deprecation issues, and performed extensive tests to improve upon the Pearson correlation values on our test set. By increasing the number of epochs our baseline model was trained on, as well as changing the hidden size of the BiLSTM used, we were able to increase the Pearson correlation value on the test set from 0.6825 up to 0.8556. This comes close to the Pearson correlation values that Brahma received with the original REGMAPR model on their test set. After testing with multiple optimizers, we decided the optimizer that gave the best performance was ADAM, which was used in both the baseline model and the model Brahma proposed. Alternative optimizer options are Adamax and Adadelta, which both performed slightly worse than Adam, but still gave good results. We also tried using cross entropy loss instead of mean squared error, but since our model only has one class, the relatedness score, it did not perform well. When using REGMAPR for textual entailment tasks with multiple classes, such as the SNLI dataset, which has labels for “contradiction,” “neutral,” and “entailment,” cross entropy loss can be used as an alternative to mean squared error loss.

If we were given more time, more team members, or more computational resources to further improve upon our project, our first step would be to implement use of the SNLI dataset for textual entailment [6]. Through the use of this dataset, we could compare our test accuracy with that of the accuracy for REGMAPR published on the SNLI corpus. Since we did not have time to make use of the SNLI dataset we focused our efforts on using the SICK dataset and sentence relatedness. If given more time, we could thus make use of the SNLI dataset and see accuracy as opposed to Pearson correlation values. From there, we could continue to adjust and test parameters and other features to see if we could further improve REGMAPR’s performance. If given more time, another improvement our group would pursue would be to fully implement the paraphrase feature to use the PPDB. Our model currently checks the dataset for paraphrased words, and not the PPDB, leaving out many potential paraphrases found in the PPDB. Having a full implementation of the paraphrase feature would further increase the Pearson correlation values, getting us closer to the Pearson correlation values that Brahma received. With more time, we would have also translated our model to an ipynb, as currently our model is not in a python notebook and instead is just composed of multiple python files. Translating the code to an ipynb would allow us to visualize different aspects of our model and find further ways of increasing the Pearson correlation values received, as well as decreasing overfitting present.

**References**

[1] Brahma, S. (2018). REGMAPR-Text Matching Made Easy. *arXiv preprint arXiv:1808.04343*.

<https://arxiv.org/pdf/1808.04343v1.pdf>

[2] Wang, S., & Jiang, J. (2016). A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

<https://arxiv.org/pdf/1611.01747.pdf>

[3] He, H., & Lin, J. (2016, June). Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 conference of the north American chapter of the Association for Computational Linguistics: human language technologies* (pp. 937-948).

<https://www.aclweb.org/anthology/N16-1108.pdf>

[4] Liu, Y., Sun, C., Lin, L., & Wang, X. (2016). Learning natural language inference using bidirectional LSTM model and inner-attention. *arXiv preprint arXiv:1605.09090*.

<https://arxiv.org/pdf/1605.09090v1.pdf>

[5] Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

<https://arxiv.org/pdf/1606.01933.pdf>

[6] Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

<https://nlp.stanford.edu/projects/snli/>

[7] Pavlick, E., Rastogi, P., Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2015, July). PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 425-430).

<https://www.aclweb.org/anthology/P15-2070.pdf>

[8] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

<https://www.aclweb.org/anthology/D14-1162.pdf>

[9] Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., & Zamparelli, R. (2014, May). A SICK cure for the evaluation of compositional distributional semantic models. In *Lrec* (pp. 216-223).

<http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf>

[10] Mueller, J., & Thyagarajan, A. (2016, March). Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 30, No. 1).

<https://people.csail.mit.edu/jonasmueller/info/MuellerThyagarajan_AAAI16.pdf>

[11] Cunningham, Ben. (2018). regmapr, GitHub repository.

<https://github.com/benjcunningham/regmapr>

[12] Li, B. (2017, November 22). *Strange behavior with Adam optimizer when training for too long*. Data Science Stack Exchange.

https://datascience.stackexchange.com/questions/25024/strange-behavior-with-adam-optimizer-when-training-for-too-long

[13] Brownlee, J. (2019, August 6). *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*. Machine Learning Mastery.

<https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>

**7. Contributions**

Dylan Sholtes – Ran all the tests and gathered all the results. Fixed all deprecation issues present with the model. Helped write the models, future, features, and discussion sections in the Final Poster. Helped write the discussion and conclusion and future work sections in the final report. Helped with editing throughout and formatting of the Final Report document.

Christie McDaniel – Made shared documents for proposal, poster, and reports, and edited all documents. Formatted the Project Poster as well as the Final Report document; designed the Project Poster. Helped write the motivation and data sections in the Final Poster and formatted the table and chart of results. Helped write the abstract, results, and conclusion and future work sections in the final report. Organized and cited all references in APA style.

Ruslan Cahnovsky – Assisted with changes to the model. Assisted with implementing the ability to measure the accuracy in the model. Helped write the introduction and discussion sections in the final report. Helped with proofreading the entire report and formatting.

Tyler Norton – Assisted in the examinations of the model and especially managing the changes made when switching from SNLI to SICK dataset. Helped editing throughout and with writing and proofreading documentation specifically with the dataset, methods, and discussion sections.