Metaphor Masters: LSTM and BERT for Metaphor Detection

Jake Zweifler **Owen Karpf** **Wesley Chen**

Abstract

Metaphor detection and interpretation is an ongoing task that is necessary for computers to fully understand the human language. We compare two methodologies to approaching the problem of metaphor detection across a sequence of words. First, we recreate a previous study that used bi-directional LSTMs, alongside ELMo and GloVe vectors, to detect metaphors (Gao et al., 2018). Second, motivated by the historical outperformance of BERT compared to LSTM based models across a variety of tasks, to we propose and apply BERT and DistilBERT, to the problem of sequential metaphor detection. After experimentation, we find that BERT outperforms Long-Short Term Memory Networks (LSTMs) in accuracy across all of the TroFi, MOH-X and VUA datasets. It performs only slightly worse in F1 for MOH-X and VUA but performs significantly worse in F1 for TroFi.

1 Introduction

In their seminal work, linguists Lakoff and Johnson (1980) define metaphor as a “mapping between a source and target domain”. In the study of linguistics, there are two different notions of metaphor — conceptual metaphors and linguistic metaphors. The study of conceptual metaphors looks at the metaphysical mappings between concepts while linguistic metaphor refers to specific instances of metaphor in text. Although conceptual metaphors are important to the linguistics field, the study of conceptual metaphors shines little light on the meaning of specific metaphors within individual sentences. Since the intention of this paper is to add to the existing literature about detecting metaphors within text, this paper solely looks at detecting linguistic metaphors.

As a task, metaphor detection is essential for computers to fully understand text. Whether it be machine translation, sentiment analysis or language generation, metaphor detection is essential to achieving correct outputs with models that fully understand the human language. Indeed, without the ability to detect metaphors and correctly interpret them, NLP models will never fully succeed at understanding sentences like "some nice results emerged from the study," in turn limiting their performance on the aforementioned tasks.

We look into approaching metaphor detection with two different machine learning models. We first apply bi-directional LSTMs to the problem. LSTMs are a version of a recurrent neural network that is modified with the intentions of limiting problems such as vanishing and exploding gradients and enabling better retention of hidden states and inputs that are more relevant to the final output. Second, we propose DistilBERT as a novel way to detect metaphors within a sequence of words. All relevant code for this paper can be found at https://github.com/wesleytchen/nlp-finalproject.

2 Overview of the Models Used

2.1 LSTM Model

LSTMs are deep neural networks that build on the architecture of a recurrent neural network, where hidden states of previous word embeddings (in the case of metaphor detection, previous inputs would be previous word embeddings in the sequence) are used as as part of the inputs for the hidden state of the next word embedding in the sequence, with the intention of determining contextual information for a given wSord in a sequence. However, recurrent neural networks quickly run into the problem of vanishing gradients when used on larger sized input sequences, as hidden states repeatedly get squashed together when activations functions like sigmoid or tanh are used. LSTMs were created to address this problem. They do this in a couple of ways. For one, unlike in recurrent neural networks, where the hidden state of the previous word is fed directly into the activation function of the next word (along

with other inputs), in LSTMs, the hidden state of the previous word is added to the result of the activation function of the current word embedding. Additionally, LSTMs add input, output and forget gates to the neural architecture with the intention that these gates are able to determine which inputs and hidden states are important.

In addition to simple LSTMs, there are also more complex models built on this initial architecture. One important modified LSTM model that is frequently used is the bi-directional LSTM. While a typical LSTM typically only looks at a sequence of words along one direction (e.g. left-to-right or right-to-left) and one layer, a bi-directional LSTM typically uses two LSTM layers stacked on top of each other, with the outputs of each layer for each word embedding being combined together in some manner (e.g. concatenation or averaging) in the end. These two LSTM layers that are stacked on top of each other also run in different directions, with one going left-to-right and the other going right-to-left. The intention of this bi-directioanlly is to determine contextuality for a given word in a sequence from both the words that came from before it and from the words that came after it.

2.2 BERT

BERT is a transformer based model for natural language processing related tasks. It is a newer alternative to LSTMs which was created and published by Google in 2018. BERT pre-trains bidirectional representations of unlabeled text by conditioning that representation on the left and right context words. One advantage of BERT is parallelism because it uses one layer to perform all of the training and one additional output layer. The disadvantage to this approach is that the window size is fixed and only the immediate context words are used, as opposed to the unbounded memory of LSTMs. An analysis of BERT in 2020 concluded that it forms a new baseline for natural language processing tasks.

At the core of BERT are transformers. Transformers encode text similarly to recurrent neural networks but they do not include recurrent modules. This means that they do not encode text sequentially and thus there is no sequential dependency. Instead, transformers rely on attention only which are weights used to average features captured in the input embedding.

3 Literature Review

3.1 Previous Approaches using LSTM

As an ongoing area of research, the question of metaphor detection has been approached using a wide variety of modern machine learning methods. In the main reference for this paper, Gao et al. (2018) use a bi-direction long short-term memory network (LSTM) in tandem with a simple feed-forward neural network to determine the metaphoricity of target verbs within a sequence. Specifically, Gao et al. begin with pre-trained word embeddings that are concatenated with ELMo based word vectors. Additionally, the target verb also has an index embedding concatenated to it in order to label it as the target verb. These word embeddings are then fed to a simple bi-directional LSTM, where the left-to-right LSTM layer begins at the first word in the sentence and ends at the last word, while the right-to-left LSTM layer begins at the last word in the sentence and ends at the first word in the sentence. Working in tandem with the bi-directional LSTM layers is an attention attention layer, and the results of the LSTM layer and the attention weights from the attention layer are then multiplied and summed together for all words in the sequence. This result is then passed through a simple feed-forward network in order to get the final output for the target verb within the given sequence.

Swarnkar and Singh (2018) also look at the problem of metaphor detection using bi-directional LSTMs, but use a significantly modified approach. Like Gao et al., Swarnkar and Singh also use bi-directional LSTMs. However, in addition to not concatenating ELMo based word vectors to their pre-trained word embeddings, Swarnkar and Singh's approach differs from that of Gao et al., in that they split the sequence of initial word embeddings into two sets of word embeddings. These two sets are $A = \{w_1, ..., w_i\}$ and $B = \{w_n, w_{n-1}, ..., w_i\}$, where w_1 is the word embedding for the first word in the sequence w_i is the word embedding for the target verb and w_n is the word embedding for the final word in the sequence. Set A is then fed to a left-to-right LSTM, while set B is fed to a right-to-left LSTM. The results of these two LSTM layers are then concatenated together. Additionally, Swarnkar and Singh take advantage of the idea that the difference between a verbs literal meaning and it's contextual meaning impact the probability of it being metaphorical

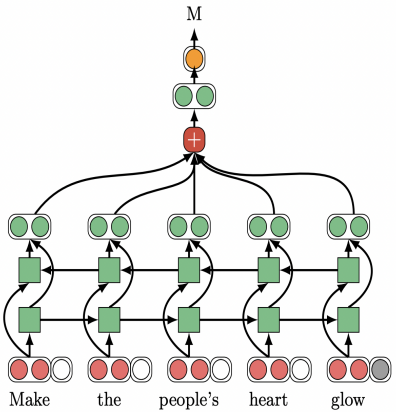

Figure 1: Architecture for the classification Bi-Directional LSTM model we recreated. Adapted from Gao et al.

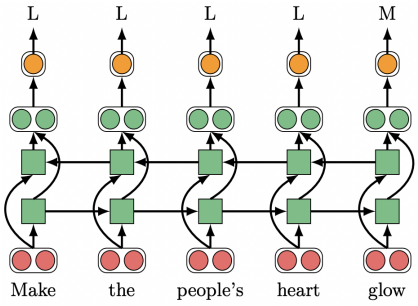

Figure 2: Architecture for the Classification Bi-Directional LSTM model we recreated. Adapted from Gao et al.

by creating a final feature vector of the word embedding for the target verb concatenated with the results of the LSTM subtracted from the word embedding for the target verb. This final feature vector is then fed to a simple feed-forward neural network to determine the metaphoricity of the target verb in the given sequence.

A simpler approach to metaphor detection is taken by Do Dinh and Gurevych (2016) who combine word embeddings with neural networks. The specific neural network used is a multi-layered perceptron (MLP) and a feed forward neural network. In this way, the neural network is simpler than in the previous two works since it is not LSTM. Nonetheless, the classification task is quite similar in that inputs are represented by word embeddings and classified as either literal or metaphorical. Another difference is that every input token is given such a classification, not just specific verbs. Thus, the paper is more general, focusing on classifying every part of speech. The specific word embedding used is a concatenation embedding with window size 5. Specifically, the input to the neural network looks up pre-trained word embeddings for each input token and concatenates the 5 word embeddings. This then becomes the input for the neural network. The pre-trained word embeddings are the 300-dimensional embeddings created by Google's word2vec.

3.2 Previous Approaches using BERT

BERT-based approaches have also been tried. Choi et al. (2021) apply RoBERTa, a more robustly trained version of the original BERT model, to the classification task of metaphor detection. Along with RoBERTA as a backbone of their architecture, they combined two different methods of recognizing metaphors. Firstly, they looked at whether or not the target word's meaning within it's given context was different from it's literal meaning. Secondly, they used the idea that a target word is more likely to be metaphorical if the other words within the sequence rarely accompany the target verb.

Similarly, Maudslay et al. (2020) use BERT large. However, along with BERT large, they use an ensemble-model that incorporates a deep MLP network. Specifically, Maudslay et al. use BERT to learn contextualized word embeddings, which alongside static Word2Vec word embeddings and vectors indicating concreteness are fed into the MLP. The model is ensemble-based as it backpropogates through BERT, meaning that the BERT-based embeddings are learned as the model trains.

4 Our Approach

4.1 Baseline:

We used the same baseline as found in the study done by Gao et al. Specifically, we classify a target verb as metaphorical if the target verb is labeled as metaphorical more frequently than it is labeled as literal. For example, if the word "flew" was labeled as a metaphor twice and only labeled as literal once, the sentence "He flew away" would be classified as metaphorical.

4.2 Classification vs Sequence Tasks

The task of detecting metaphors can be viewed in two ways. Firstly, we can view it as a task in which we are classifying a single target word as metaphorical, and from there, deriving the metaphoracity of

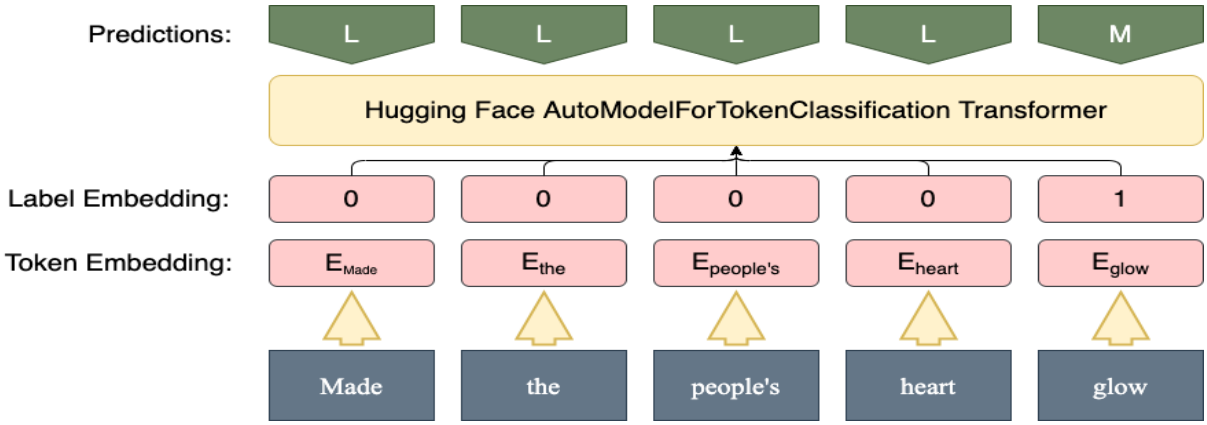

Figure 3: Architecture for the DistilBert and BERT Models using Hugging Face's automatic model for classifying tokens within a sequence

the entire sentence. Alternatively, we can view the task as the classification of each word in a sequence as metaphorical or not. As Gao et al. noted, the latter task generalizes the first task, as when each word in a sequence is assigned a label of metaphorical or not, the target word within the given sequence will also be assigned a label of metaphorical or not.

Since Gao et al. implement both of approaches, we recreate both tasks with a bidirectional LSTM model. Importantly, the inputs for each task are slightly different. Specifically, for the classification task, within each sequence, an embedding is concatenated to the target verb. For the sequential task, labels of each word are used as part of the input. However, because the TroFi and MOH-X datasets only contain metaphoracity labels for the target verb, we use the same assumption that Gao et al. used, namely that words that are not labeled as metaphorical are literal.

With DistilBERT and BERT however, we only implement the sequential task, as the classification task of the target verb is ingrained within the sequential task.

4.3 DistilBERT vs. BERT

We implemented two BERT-based models for the task of sequential metaphor detection. The architecture for the models are identical, but the model itself is changed. We also trained BERT on 10 epochs for VUA and 50 epochs for MOH-X because we saw better performance over DistilBERT after training these models with fewer epochs. Specifically, BERT was trained on 10 epochs for VUA and 50 epochs for MOH-X while DistilBERT was trained on 5 epochs for VUA and 10 epochs for MOH-X.

DistilBERT is a smaller and lighter version of BERT that is faster to train for a particular task (Sanh et al. 2019). Compared to BERT, DistilBERT is 40% smaller, 60% faster, and retains 97% of language understanding capabilities. It does this through distillation, which is a compression technique wherein a student model learns to replicate the results of a teacher model. In this case, the teacher model is BERT while the student model is DistilBERT. We choose to use both BERT and DistilBERT because they are both supported by HuggingFace's AutoModel (Hugging Face).

4.4 Data Set

For all tasks, we used three datasets. The TroFi dataset contains samples of text containing literal and nonliteral usages of 50 English words. It contains 3737 sentences, with 43% of sentences within the dataset containing target verbs that are metaphorical. Samples come from The 1987-89 Wall Street Journal (WSJ) Corpus Release 1 and the dataset is publicly available for download from SFU LangLab. The MOH dataset is a subset of the WordNet corpus where verbs have been tagged for metaphors using the crowd-sourcing website CrowdFlower. The MOH-X dataset is a subset of the MOH dataset that only contains the verb–direct object and verb–subject relations of the annotated verbs (i.e. instances with pronominal or clausal subject or object are not used). Within the MOH-X dataset, there were 647 sentences, of which 49% were metaphorical. As previously mentioned, both datasets only contain literal and non-literal labels for the target verb, with the other words being left unlabeled.

We also used the VUA dataset. This is a publicly available dataset of academic texts, news texts, fiction, and conversations that are hand-annotated for metaphors regardless of lexical field or source domain. This makes the dataset ideal for tagging because there is no discrimination by part-of-speech. In other words, each word within a given sequence is assigned a label of metaphorical or not, making it perfect for the sequential task. Furthermore, unlike the TroFi and MOH-X datasets, the VUA dataset comes already broken up into train, test and validation data sets. The dataset as a whole had 23113 sentences, of which 28% were metaphorical.

Note that some datasets are unbalanced with more metaphors than non-metaphors. The TroFi and MOH-X datasets in particular are purposefully constructed to have more metaphors than in naturally occurring text. On the other hand, the VUA dataset is collected from general text that is annotated post-hoc, so it is more likely to be a balanced corpus.

5 Results

5.1 LSTM Sequence and Classification Results:

The results of the bi-directional LSTM are nearly identical to those found by Gao et al. Much like Gao et al, the recreated Bi-directional LSTM outperformed the baseline model on all measurements besides the TroFi precision.

However, though similar for the most part, there were a few major discrepancies between the findings of Gao et al. and our reproduction of their findings. Specifically, the MOH-X recall we find when recreating the study is nearly seven points of from recall that Gao et al. found. Furthermore, the precision, recall and accuracy of the VUA classification task was significantly different than what Gao et al. found when we recreated it. In fact ,our accuracy for the task was over 12% higher than what was found in the paper. It's unclear why this discrepancy exists, since the discrepancy is too large to be due solely to the mild stochastic nature of NLP models.

Overall though, our reproduction of the study conducted by Gao et al. demonstrates that the bi-directional LSTM performs relatively well across the board at detecting metaphors.

5.2 BERT Results:

For all three data sets, BERT achieves higher accuracy than LSTM. The accuracy is 22.05 points higher for MOH-X, 25.51 points higher for TroFi, and 13.35 points higher for VUA. For MOH-X and VUA, the F1 is only slightly worse, with a 1.44 drop in F1 for MOH-X and a 0.6 point drop in F1 for VUA. However, there is a large 13.25 drop in F1 for TroFi.

Additionally, BERT outperforms DistilBERT in all data sets but TroFi. In that case, DistilBERT has 2.23 points higher F1 with the same accuracy. Qualitatively, TroFi is different from VUA and MOH-X because it has a small set of target verbs in different contexts while the other data sets have a variety of potential metaphors that we can train on. We also trained BERT using holdout datasets rather than 10-fold cross validation as was used for LSTM. This could make it harder for BERT to capture general information about the verbs due to the lack of inherent variety.

One potential issue we examined was that BERT uses more granular tokens than LSTM. This means that the model might result in a higher level of false negatives because a single metaphor might be split into several parts where only one part is accurately classified. To analyze this error, we reconstructed the original tokens and kept the metaphor label for each token if any of the subtokens were labeled as metaphor. However, this process did not increase recall or prevision for any data set except for VUA, where precision increased from 70.84 for BERT to 72.33 for BERT with detokenization. However, the F1 score did not increase so the tradeoff between increased precision and decreased recall did not lead to better overall model performance. These results show that the granularity of the tokens was not a major contributor to the decreased F1 performance of BERT compared to LSTM.

For our Parts of Speech analysis of BERT, we saw that BERT had similar F1 score to LSTM for all parts of speech except nouns and adpositions. For nouns, BERT had 11.89 points higher F1 while for adpositions, BERT had 26.34 points lower F1. However, since there are almost twice as many nouns in the sample than adpositions, the overall F1 for BERT and LSTM is similar.

5.3 Sentence Level Error Analysis

Here are three sample sentences where we used LSTM to analyze a sequence of words for sen-

POS	#	% metaphor	LSTM			DistilBERT			BERT		
			P	R	F1	P	R	F1	P	R	F1
VERB	20K	18.1	68.32	70.64	**69.46**	67.92	63.90	65.85	70.54	65.38	67.86
NOUN	20K	13.6	66.22	58.02	61.85	75.14	70.65	72.83	77.12	70.65	**73.74**
ADP	13K	28.0	87.86	88.38	**88.11**	63.12	60.30	61.68	63.05	60.55	61.77
ADJ	9K	11.5	63.65	58.60	**61.02**	61.30	50.00	55.08	62.37	52.30	56.89
PART	3K	10.1	60.53	61.75	**61.13**	51.95	49.59	50.74	59.92	59.92	59.92

Table 1: Performance of the Baseline, Classification, Sequence and BERT models

Model	MOH-X				TroFi				Vua			
	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1	Acc.
Baseline	39.09	26.70	31.29	43.56	**72.39**	55.74	62.92	71.42	67.90	40.72	50.91	76.45
Classification	**78.46**	**77**	**77.33**	77.89	68.45	**76.53**	**72.19**	74.26	72.27	61.73	66.58	81.42
Sequence	74.73	75.37	74.54	75.16	68.33	73.29	70.51	73.38	68.51	**69.79**	**69.14**	81.32
DistilBERT (Seq)	58.39	55.41	56.86	94.90	60.73	58.30	59.49	**98.89**	67.77	64.38	66.03	94.12
BERT (Seq)	76.81	69.74	73.10	**97.21**	63.27	52.29	57.26	**98.89**	70.84	66.38	68.54	**94.67**
BERT with Detokenization (Seq)	73.81	61.18	66.91	96.20	59.83	54.48	57.03	98.72	**72.33**	64.21	68.03	94.42

Table 2: Performance of the Baseline, Classification, Sequence and BERT models

tences in the VUA dataset. Bolded words are metaphors, words underlined in blue are false positive, and words underlined in red are false negatives:

1. Other British towns , Croydon and Southampton **among** them , are also **considering** modern tramways .

2. Dead , ragged **heads** of the **climbing** hydrangea can be removed .

3. If , **on** the other **hand** , you **allow** rationality **to** children , you can't use their lack of it as criterion to **distinguish** them **from** adults .

Here are the same sentences where analysis is done by normal DistilBERT:

1. Other British towns , Croydon and Southampton **among** them , are also **considering** modern tramways .

2. Dead , ragged **heads** of the **climbing** hydrangea can be removed .

3. If , **on** the other **hand** , you **allow** rationality **to** children , you can't use their lack of it as criterion to **distinguish** them **from** adults .

Here are the same sentences where analysis is done by normal BERT:

1. Other British towns , Croydon and Southampton **among** them , are also **considering** modern tramways .

2. Dead , ragged **heads** of the **climbing** hydrangea can be removed .

3. If , **on** the other **hand** , you **allow** rationality **to** children , you can't use their lack of it as criterion to **distinguish** them **from** adults .

These examples reflect the data found in that BERT and DistilBERT both seem to be more conservative than LSTM. This explains why the recall values are lower than the precision values for many of the tests. Reading the sentences, it certainly makes sense how the models missed the metaphor words that were missed. For example, the word "climbing" refers to the "hydrangeas," a concept that the model may not recognize. Checking the dataset, this word is only used in two sentences, so the model would have no way of knowing that the flowers could not literally climb. This explains why "climbing" is missed by all three models.

6 Discussion and Conclusion

The performance of BERT and LSTM is most accurately characterized as a tradeoff between F1 and accuracy. BERT has higher accuracy than LSTM for all data sets but worse F1. For a class-imbalanced data set such as the ones analyzed in this paper, F1 is a preferred metric over accuracy. However, for VUA and MOH-X, the drop in F1 for using BERT is less than two points while the increase in accuracy is more than 10 points. This suggests that despite the lower F1 score, BERT

has superior performance for these data sets when compared to LSTM.

6.1 Future Possibilities:

Going forward, there are a couple of additional directions we could take to improve our model. For one, we can attempt to apply other BERT-based models, like RoBERTa, which is a more robustly trained version of BERT. Additionally, we can try to apply metaphor detection to other realms of NLP, like text generation. Specifically, we think it would be interesting to combine metaphor detection with metaphor interpretation in order to suggest words with literal meaning that could replace the metaphorical word.

7 Extra Comments about Metaphor Detection

After completing the project, we now consider some more intangible questions for why metaphor detection is an interesting topic to consider in the first place!

7.1 Importance of Metaphor Detection in NLP

Metaphors are present in almost all types of writing — a fact that is inherent in the flexibility of humans and our language. Thus, in order to fully use natural language processing to analyze text, it is crucial to account for non-literal usages of words. Other reason that metaphor detection is quite useful is for text generation. Although our task involved classification, one day, we may hope to create robots that talk identically to humans. Of course, there are programs that can pass the Turing test, but these robots are still incapable of holding advanced conversations about elevated topics. Since humans use many metaphors, especially in colloquial speech, understanding metaphors is quite important. The first step to metaphor generation is metaphor detection.

7.2 Metaphor Detection is Inherently Difficult

As mentioned, a metaphor is just a mapping between similar concepts. So certainly, many types of these mappings exist. This is one reason why metaphor detection can prove to be specifically difficult, since our models may need to look for many different things at the same time. For example, personification is very different than direct metaphor. We didn't define these terms in our paper, since we just considered all metaphors to be the same, so here are examples:

- They danced into my heart (direct metaphor).
- It danced across the sky (personification).

Note that the reason "danced" is a metaphor is different in both of these sentences. In the first case, the model must recognize that it is impossible to literally dance into one's heart. In the second sentence, the model must recognize that "it" refers to an outer-space object that is not alive.

Another reason that metaphor detection is difficult is that annotating the data set is often more difficult than most other NLP tasks. For the reasons above, it already hard for humans to detect metaphors. There are many cases in the data sets where we disagreed with the labels given. Therefore, in addition to obtaining a large enough data set, finding a well annotated and consistent data set is even harder.

8 Personal Evaluation

8.1 Challenges Faced

We had to get the data from the authors, since they used a specially formatted and truncated version of the publicly available data sets. They also used embeddings that are not obtainable. Though the code for the project was on their github, the data was only available upon request. As a result, we were not able to acquire the data right away, slightly delaying us. Fortunately, the authors sent us the data and we were able to proceed with the model.

Another challenge we faced was that the data sets were not formatted for BERT. We wrote a python script to extract tokens and label embeddings from unformatted raw text.

At some points, we got frustrated with the project due to issues with our BERT model and with formatting our documents. However, we kept our spirits high by encouraging each other.

8.2 How Well We Did

We met all the requirements of the final project. We were able to replicate all of tables 5 and 6 in Gao et al. (2019). We also applied BERT to the experiments at hand by building a BERT model to perform the sequence task. This model was also used to find Part of Speech specific metrics. Since our classification version of BERT would

have the same architecture as our sequence version, we chose not to evaluate this task with BERT.

We went above the minimal requirements by exploring possible reasons for the performance metrics we found using BERT. We tested the possibility that token size negatively impacted our results by reconstructing the original tokens from the output of BERT and found that token size did not explain why BERT underperforms LSTM for certain metrics. We also tested both DistilBERT and BERT to see if either model would be better.

Our findings for BERT show that we were able to demonstrate some improvement over the original findings in Gao et al. First, the accuracy for BERT was higher for all three datasets. We also had only minimal decrease in F1 score for VUA and MOH-X. In the parts of speech analysis, we had similar F1 for all parts of speech but we also had higher F1 for Nouns.

8.3 Group Member Contribution

We all worked quite well together, splitting up the work effectively while still all helping on each part. Specifically, Wesley did most of the coding, Owen did most of the research, and Jake did a lot of the poster/paper writing. Of course, we all helped on all three parts, but these were the areas in which we specialized. Wesley, especially, did a great job running many different types of BERT models on the last day after getting some feedback at the poster conference.

Acknowledgements

Thank you very much to Mourad Heddaya, who provided last minute office hours help. We'd also like to thank Ge Gao, the first author of the "Neural Metaphor Detection in Context" paper, for sending us all of the cleaned datasets that they used in their paper, as well as the pre-trained ELMo vectors that they created.

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 329–336, Trento, Italy. Association for Computational Linguistics.

Julia Birke and Anoop Sarkar. 2007. Active learning for the identification of nonliteral language. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 21–28, Rochester, New York. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. Association for Computational Linguistics.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 28–33, San Diego, California. Association for Computational Linguistics.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613, Brussels, Belgium. Association for Computational Linguistics.

Sylvain Gugger. 2021. Token classifcation example.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and wordnet based metaphor identification and interpretation. In *ACL*.

Saif Mohammad, Ekaterina Shutova, and Peter Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33, Berlin, Germany. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. volume 8, pages 842–866, Cambridge, MA. MIT Press.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 160–170, San Diego, California. Association for Computational Linguistics.

Gerard Steen, Lettie Dorst, J. Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*.

Krishnkant Swarnkar and Anil Kumar Singh. 2018. Di-LSTM contrast : A deep neural network for metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing*, pages 115–120, New Orleans, Louisiana. Association for Computational Linguistics.