Transformer based metaphor detection

https://github.com/shash4897/Metaphor-Detection-NLP

Raksheet Bhat RRR200001 | Jayen Boopathy JXB200032 | Shashank Saran SXS210186 | Nitishree Supekar NXS210021

1 Introduction (5pt)

Natural language frequently uses metaphors and recognizing them requires making rigorous judgments about the chance that specific circumstances may happen. Here, we offer complete neural metaphor detection models that can recognize certain word representations in context, which are crucial for accurately understanding figurative language. Unlike prior techniques, which focused on certain forms of linguistic context, our models consider the full language context. Predictions get more precise, therefore.

Our study covers two commonly used task formulations: detecting whether a target verb in a phrase is metaphorical or not and recognizing all metaphorical words in each sentence. Using the comprehensive language context, we proposed end-to-end neural models that outperform prior models based on confined linguistic context. These tasks are also subjected to transformer-based models, and their performance (F1, precision, and recall metrics) is contrasted with that of Bi-LSTM models.

2 Task (5pt)

Sequence labeling and classification are the two formulations of the metaphor detection task. The goal of sequence labeling is to determine if each word in a given phrase (x1,..., xn) is metaphorical or not, and to forecast a sequence of binary labels (l1,..., ln) stating as much. When classifying a sentence, the objective is to predict a binary label (l) that indicates the metaphoricity of the target word (xi), which is located at a predetermined index (i). While both tasks have been studied before, it is important to remember that the sequence labeling task also includes the classification task because the target verb prediction can be inferred from the predictions for the entire sentence. The sequence labelling task is akin to a part-of-speech tagging problem where your only labels for each token are 'literal' and 'metaphor'.

3 Data (5pt)

The VU Amsterdam Metaphor Corpus is by far the largest annotated dataset available for metaphor detection tasks (some of the others we saw were MOH and TroFi). Due to its size, availability, reliability of annotation, the VUA dataset has been popularly used for training with many supervised learning-based algorithms, and for this reason, we also decided to use this dataset for our task. The dataset consists of 117 fragments sampled across four genres from the British National Corpus: Academic, News, Conversation, and Fiction. Each genre is represented by approximately the same number of tokens, although the number of texts differs greatly, where the news archive has the largest number of texts. We randomly sampled 10% of the data for our validation set, 10% for test set and the remaining 80% for our training set.

It was a challenge to get the VUA dataset - it was not available on their official website [7] due to licensing restrictions. However, a CodaLab competition "Metaphor Shared Task" [8] provided a script [9] to parse the original VUAMC.xml to extract the verbs and other content words required for the metaphor detection task. We ran this script to obtain the VUA dataset for the sequence labelling task.

4 Methodology (20pt)

4.1 Bi-LSTM Model

This project focuses on using a Bidirectional LSTM as the baseline model. A BiLSTM is a type of recurrent neural network (RNN) architecture that includes two LSTM (Long Short-Term Memory) layers, one processing the input sequence in

	Train	Dev	Test	Total
#tokens	150,056	19,757	17,570	181488
#unique tokens	15,567	3,754	2,530	17,874
#sentences	9687	1211	1211	12109
%metaphor	6.2	6.6	7.4	6

Table 1: VUA dataset statistics

a forward direction, and the other processing it in a backward direction. This allows the network to capture both past and future contexts of a given input sequence.

We use this BiLSTM to tag the tokens as metaphor or literal. Specifically, this model takes as input a batch of text sequences, represented as integer indices, and outputs a set of predicted labels for each token in the sequence.

The architecture consists of four main components:

Embedding Layer: This layer maps each input token to a low-dimensional dense vector representation (embedding) using an embedding matrix. The reference paper [1] uses GloVe and ElMo embeddings. However, we found that in comparison to GloVe, the FastText gives better results.

BiLSTM Layer: This layer processes the input sequence in both forward and backward directions using an LSTM architecture. The output of this layer is a sequence of hidden states, where each hidden state captures contextual information for the corresponding input token.

Dropout Layer: This optional layer randomly drops out a fraction of the output of the BiLSTM layer during training to prevent overfitting.

Dense Layer: This layer maps the output of the BiLSTM layer to the final output space (signifying if a token is a metaphor or literal) using a linear transformation.

During training, the model is optimized using the cross-entropy loss function(weighted) and the Adam optimization algorithm. During inference, the model takes a text sequence and its length as input, and produces a set of predicted labels for each token in the sequence by performing a forward pass through the model. The predicted labels can be obtained by selecting the label with the highest probability for each token in the output sequence.

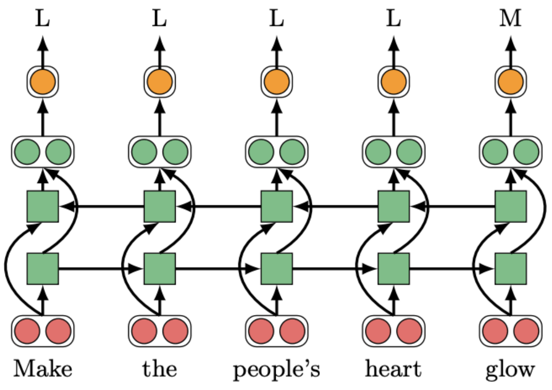

Figure 1: A sequence labeling Bi-LSTM model for metaphor de- tection. Every word in a sentence is classified

4.2 Transformer based models

Transformer models have quickly become the state-of-the art methods for many common natural language tasks such as text classification, summarization, question answering etc. The key advantage they have over their prior counterparts is their ability to infer long distance relationships between tokens by being completely attention based. Additionally, many modern transformer based models have been pretrained on large corpuses of data to be able to generate contextual embeddings for tokens. For this reason, we decided to finetune several transformer based models on the sequence labelling formulation of the metaphor detection task and to observe if these models perform as well on this task as they do on other natural language tasks. For our project, we decided to use three such models.

[11]BERT was our first model and is one of the most popular transformer models, developed by Google. It is an encoder only transformer model which is pretrained on the masked language modelling and the next sentence prediction tasks and is capable of being finetuned on several downstream natural language tasks such as token classification, question answering and natural language inferencing. It uses Wordpiece tokenization strategy which is a subword tokenization strategy.

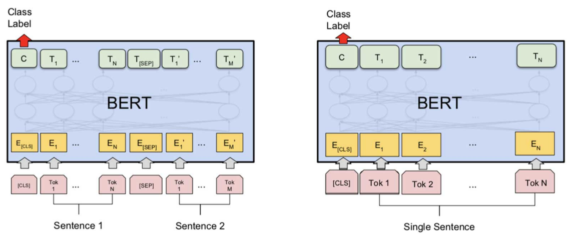

Figure 2: The BERT model - architecture and pretraining tasks

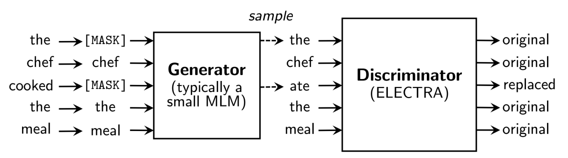

Figure 3: The ELECTRA model - architecture and pretraining tasks

Input embeddings consist of token embeddings, segment embeddings and position embeddings.

[12]Our second model is the RoBERTa model developed by Meta (earlier Facebook). This model is variation of BERT and was designed to improve upon the BERT model. RoBERTa gets rid of the next sentence pretraining objective employed by BERT and focuses solely on the masked language modelling task. Another key difference between RoBERTa and BERT is that it uses dynamic masking rather than the static masking performed by BERT. This means that sentences are randomly masked at training time rather than being masked before training begins. Thus the masked variations of sentences are limitless allowing for more robust contextual information. To further bolster its performance, RoBERTa was trained on 10 times more training data than BERT. RoBERTa also uses Byte Pair Encoding for it's tokenization strategy instead of Wordpiece.

[13]Our last model in consideration is the ELECTRA model designed by google to overcome some shortcomings of the BERT and RoBERTa family of models. ELECTRA completely shifts the pretraining paradigm, moving away from masked language modelling task and instead opting for a more adversarial approach. It is pretrained on the task of distinguishing whether a token has been replaced in the input and to tag each token as replaced or original. The replacements are closely plausible fakes. Similar to GANs the generator tries to generate plausible fake replacements while the discriminator tries to differentiate between the replacements and the original tokens. This method results in powerful representation learning for the model.

For BERT and RoBERTa, we use the distilled versions of these two models. Distillation is the process of training a small student model to mimic a larger teacher model as close as possible. These are smaller models that train much faster but capture most of the representation power of these models.

5 Implementations (15pt)

5.1 Bi-LSTM Model

The input to the model is a list of data instances, each of which contains a sentence and a list of labels corresponding to each token in the sentence. We use a batch size of 32, number of epochs = 15, and defined a pre-calculated weighting scheme to compute the CrossEntropyLoss (0: 0.06, 1: 1.0) where 0 and 1 are the labels and 0.06 and 1.0 are the relative weights respectively. The purpose of this weighting scheme is to overcome the class imbalance while calculating loss. The embedding dimensions are 300d and the dimension of the hidden layer = 256.

5.2 Transformer based models

For our implementation, we make use of the Hugging Face[4] python library for obtaining each of the pretrained models. We used PyTorch[5] to tweak our models when required. The entirety of the code is executed on a GPU instance on Google Colab.

Preprocessing was done mainly with the pandas[14] library alongside the Hugging Face datasets library. Preprocessing steps included, lowercasing the data, tokenizing the sentences using the relevant tokenizer, aligning labels with the subword tokens and dynamically padding and batching them. We used 15 percent of the entire dataset for testing and of the remaining 85 percent, we used an additional 15 percent as our validation set.

We use a batch size = 32, learning rate = 2 * e-5, weight decay = 0.1 and number of epochs

	Bi-LSTM				BERT				RoBERTa				ELECTRA			
	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1	Acc.
literal (0)	0.92	0.80	0.85	-	0.86	0.83	0.85	-	0.98	0.76	0.86	-	0.87	0.95	0.91	-
metaphor (1)	0.36	0.35	0.36	-	0.81	0.75	0.78	-	0.91	0.79	0.85	-	0.80	0.94	0.86	-
macro avg	0.74	0.71	0.73	0.92	0.86	0.83	0.84	0.88	0.92	0.84	0.87	0.90	0.88	0.93	0.90	0.93

Table 2: Test metrics - 3 labels

= 10 while experimenting with all three transformer models. We experimented with tweaking hyperparameters but found that the training evaluation metrics did not vary too significantly and thus stuck with these.

Additionally we found that the dataset obtained had a significant class imbalance, with the amount of non-metaphor tokens being significantly greater than the amount of metaphor tokens. For this reason we had to additionally tweak the Cross-Entropy loss function to include class weights. We also included a separate label We used accuracy, precision, recall and F-1 score (macro averaged) as our metrics, implemented using Hugging Face's evaluate library.

6 Experiments and Results (45pt)

Results for both the Bi-LSTM and the Transformer based approaches can be seen in table 2. We used the following evaluation metrics - Precision, recall and F1-scores. For the Bi-LSTM model, we ran two different kinds of tests, and both cases are different in the way we deal with missing labels. In the first case, as the reference paper [1] suggested, we treated missing labels as literals. So we effectively had only 2 classes - literals and metaphors. The problem with this was that this created a huge imbalance in the dataset - there were many more literals than metaphors. To circumvent this problem, we tried a second way of treating missing labels as a third class - so we have literals, metaphors and others. This reduced the imbalance between literals and metaphors and improved our results. Below, you can find the metrics at a class level, as well as macro average scores.

	Precision	Recall	F1-score
literal (0)	0.99	0.93	0.96
metaphor (1)	0.22	0.69	0.33
macro avg	0.61	0.81	0.65

Table 3: Test metrics - 2 labels

For our transformer model, we decided to use only the three label approach as it helped ameliorate some of class imbalance problems encountered earlier. We ran the experiments with the three transformer models previously mentioned. While slightly lower and comparable in overall accuracy to the Bi-LSTM (which can be explained by the model's bias to 'other' labelled tokens), these models had significantly better per-class metrics, especially on the metaphor class. Even without class weighting, we observed that the transformers had better F-1 metrics, despite being much lower than with class weighting. Out of the three models, ELECTRA was our best performing model while BERT was the worst. This is expected as RoBERTa was designed to overcome the shortcomings of BERT, and ELECTRA was designed to overcome the shortcomings of both these models. This trend can be seen in table 1 where even though there is some variation in precision and recall, the overall and per-class F-1 score of ELECTRA is the highest (90.59%) and BERT is the lowest (84.62%).

Error Analysis For the Bi-LSTM model, we ran 2 different tests as mentioned earlier - when we had just 2 labels (literals or metaphors), there was a high class imbalance. As a result, precision of the metaphor class was very low (22%). However, when we did the analysis with 3 classes (literals, metaphor or other), the precision of the metaphor class jumped up to 36%. This was because the redistribution of classes helped overcome the data imbalance - this can also be observed from the jump in F1-score from 33% for 2 class labels to 36% for 3 class labels.

Similar to the Bi-LSTM model we found that most errors for the transformer model were on the metaphor class due to low percentage of metaphor tagged tokens.

Speed Analysis On a Google Colab T4 GPU, training time for 15 epochs for the Bi-LSTM took 6 minutes on average and for transformer models took between 13-15 mins on average.

7 Conclusion (5pt)

Using contextualized word representations, we show in this work enhanced BiLSTM models for metaphor recognition. As a consequence of our error analysis, we are able to identify the remaining challenges in metaphor identification on a number of recent benchmarks, and our models provide new state-of-the-art results. We emphasize the value of metaphor identification and provide two formulations for the problem: verb classification and sequence labeling. Using the VU Amsterdam metaphor English corpus, we conduct tests and assess the performance of multiple transformer models against a bi-LSTM model on the sequence labeling task. The transformer models significantly outperform the bi-directional LSTM model, according to our experiments and demonstrate that transformer models are capable of ably handling the metaphor detection task.

References

[1] Neural Metaphor Detection in Context (https://aclanthology.org/D18-1060) (Gao et al., EMNLP 2018)

[2] Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna A Kaal, and Tina Krennmayr. 2010. Metaphor in usage. 21(4):765796

[3] Saif Mohammad, Ekaterina Shutova, and Peter D. Turney. 2016. Metaphor as a medium for emotion: An empirical study. In Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*Sem).

[4] https://huggingface.co/

[5] https://pytorch.org/docs/stable/index.html

[6] GloVe: Global Vectors for Word Representation Jeffrey Pennington, Richard Socher, Christopher D. Manning https://nlp.stanford.edu/pubs/glove.pdf

[7] VU Amsterdam Metaphor Corpus Online! http://www.vismet.org/metcor/documentation/home.html

[8] Shared Task on VUA Metaphor Dataset https://competitions.codalab.org/competitions/17805

[9] A Report on the 2018 VUA Metaphor Detection Shared Task, Chee Wee Leong, Beata Beigman Klebanov, Ekaterina Shutova, Proceedings of the Workshop on Figurative Language Processing

[10] https://www.analyticsvidhya.com/blog /2019/11/comprehensive-guide-attention-mechanism-deep-learning/

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/pdf/1810.04805.pdf)

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach (https://arxiv.org/pdf/1907.11692.pdf)

[13] Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators (https://arxiv.org/pdf/2003.10555.pdf)

[14] https://pandas.pydata.org/