Metaphor Detection Using Context and Concreteness

Rowan Hall Maudslay[1] **Tiago Pimentel**[1] **Ryan Cotterell**[1,2] **Simone Teufel**[1]
[1] Department of Computer Science and Technology, University of Cambridge
[2] Department of Computer Science, ETH Zürich
{rh635,tp472,sht25}@cam.ac.uk ryan.cotterell@inf.ethz.ch

Abstract

We report the results of our system on the Metaphor Detection Shared Task at the Second Workshop on Figurative Language Processing 2020. Our model is an ensemble, utilising contextualised and static distributional semantic representations, along with word-type concreteness ratings. Using these features, it predicts word metaphoricity with a deep multi-layer perceptron. We are able to best the state-of-the-art from the 2018 Shared Task by an average of 8.0% F_1, and finish fourth in both subtasks in which we participate.

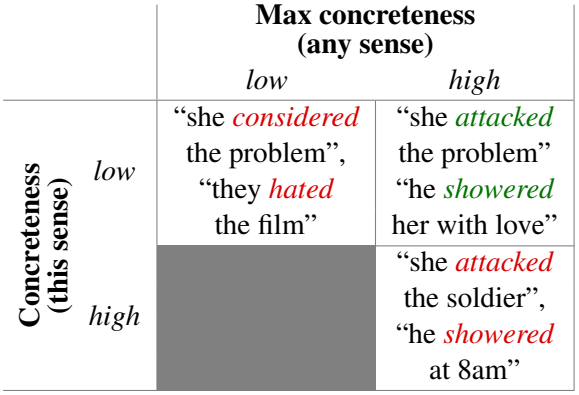

Figure 1: Examples of verbs with varying levels of concreteness—metaphors are in green

1 Introduction

Metaphor detection is the task of assigning a label to a word (or sometimes a sentence) in a piece of text, to indicate whether or not it is metaphorical. Some metaphors occur so frequently as to be considered word senses in their own right (so-called *conventional metaphors*), whilst others are creative, and involve the use of words in unexpected ways (*novel metaphors*). Sometimes whole phrases or even sentences can lend themselves to metaphorical or literal interpretations.[1] For these reasons and others, human annotators might disagree about what constitutes a metaphor—computational metaphor detection is no doubt a challenging problem.

In this work, we participate in the 2020 Metaphor Detection Shared Task (Leong et al., 2020). First, we offer a description of metaphoricity, framing it in terms of the concreteness of a word in different contexts. Concreteness of a word in context is not a quantity for which there exists large-scale annotated data. In lieu of this, we train a metaphor detection model using input features which we expect to contain the information needed to derive this contextual concreteness. This model outperforms the highest performing system of the previous shared task (Leong et al., 2018), and finishes 4th in the two subtasks in which we participate.

2 Concreteness and Context

Metaphor is a device which allows one to project structure from a source domain to a target domain (Lakoff and Johnson, 1980). For instance, in the sentence "he *attacked* the government", *attacked* can be seen as a conventional metaphor, which applies structure from the source domain of war to the target domain of argument. Intuitively, it seems that the context in which a word appears tells us about the target domain, whilst the word itself (and some knowledge about how it is used non-metaphorically) tells us about the source. Several existing models have exploited this difference (e.g. Mao et al., 2019; Gao et al., 2018).

Usually, the target domain is something intangible, whilst the source domain relates more closely to our real-world experience. *Concreteness* refers to the extent to which a word denotes something that can be experienced by the senses, and is gener-

[1] Consider *drowning student*, which could refer to students submerged in water, or students struggling with coursework (Tsvetkov et al., 2014), or the more idiomatic phrase, *they stabbed him in the back*, which could be taken literally or (more likely) metaphorically, depending on its context.

Proceedings of the Second Workshop on Figurative Language Processing, pages 221–226

https://doi.org/10.18653/v1/P17

ally measured by asking annotators to rate words on a numeric scale (Paivio et al., 1968; Spreen and Schulz, 1966); abstractness is then the inverse of concreteness. Using concreteness ratings for metaphor identification is clearly well motivated, as evidenced by previous work (e.g. Tsvetkov et al., 2014, 2013; Beigman Klebanov et al., 2015).

For a word to be metaphorical in a particular context, then, it needs to have a concrete sense and an abstract sense, with the abstract sense activated in that context. The concrete sense would belong to the source domain, and the abstract sense to the target domain. For instance, the meaning of the word *attacked* in "she *attacked* the soldier" is concrete, but in "she *attacked* the problem" it is abstract—and thus that usage is metaphorical. Polysemy of the word is a necessary condition; the existence of an abstract sense is not enough, otherwise a monosemously abstract word such as *considered* in "he *considered* the problem" would be metaphorical.

Figure 1 shows examples of words with different maximum concreteness levels elicited by certain senses (columns) appearing in contexts which result in different values of concreteness for that particular sense (rows). The most concrete sense of a word is a lexical property, and thus context independent. The metaphors (green) are found in the top right quadrant—they have an abstract meaning in context, but a concrete sense exists (as evidenced by the examples in the bottom right quadrant). The bottom left quadrant is greyed out, since it is conceptually impossible for a word to exist there—the concreteness of one sense of a word cannot be greater than the concreteness of any of its senses.

3 Model Architecture

We now describe a model which uses semantic representations of a word in and out of context to predict metaphoricity. Ideally, we would only provide the model with a representation of the concreteness of a word in context (since we believe that would do most of the lifting), but to our knowledge, no large-scale annotated datasets exist for context-dependent concreteness. In most popular datasets of concreteness annotation (e.g. Coltheart, 1981; Brysbaert et al., 2014), concreteness is a property assigned to each word type—but we would need the concreteness of a word *instance*. In this respect, our work resembles the abstractness classifier in Turney et al. (2011)—although this work uses word senses instead of instances as we do. Because contextualised concreteness data is unavailable, we instead choose features which, when given to a multi-layer perceptron (MLP), should provide enough information about a word for the MLP to be able to differentiate between each cell in Figure 1.

We first provide the model with contextualised word embeddings, which we expect will provide some information about the target domain of the metaphor. In the contextualised representations, we expect there to exist a space of concrete meanings and some space of abstract meanings—which would help the network differentiate between the top and bottom rows of Figure 1. Along with this, we provide static word embeddings, to provide information about the source domain. Since these static type-level embeddings will clearly contain information about both source and target, we compliment them with type-level concreteness ratings. Such ratings should reflect the concreteness of the most concrete sense of the word, thus allowing the network to differentiate between the left and right columns of Figure 1.

Figure 2 shows an overview of our architecture. In the following paragraphs, we detail each individual component of the model.

Contextual Word Embedding For contextualised embeddings, we fine-tune BERT (Devlin et al., 2019). BERT is a sentence encoder which utilises a transformer architecture (Vaswani et al., 2017), and is trained with two separate tasks—masked language modelling (a cloze task), and next-sentence prediction. The latent space (the final hidden state of the encoder) contains vector representations of each input token, which change in different contexts. Several pre-trained BERT models are available—we use BERT large.[2]

Concreteness Model We define a simple model which represents the concreteness of a word as a linear interpolation between two vectors, representing maximal concreteness and abstractness, $\mathbf{v}_{\text{con}}$ and $\mathbf{v}_{\text{abs}}$ respectively. For each word w we obtain a real number estimate of its concreteness, c, from Brysbaert et al. (2014), where $c = 5$ indicates maximum abstractness, and $c = 0$ indicates maximum

[2]BERT accepts WordPiece units (Wu et al., 2016) as tokens, rather than words. There is not a single accepted way of converting multiple WordPiece unit vector representations into a single word. Following Devlin et al. (2019), we simply use the first token of any word. This is clearly a naïve method of compositionality; improving this may strengthen our results.

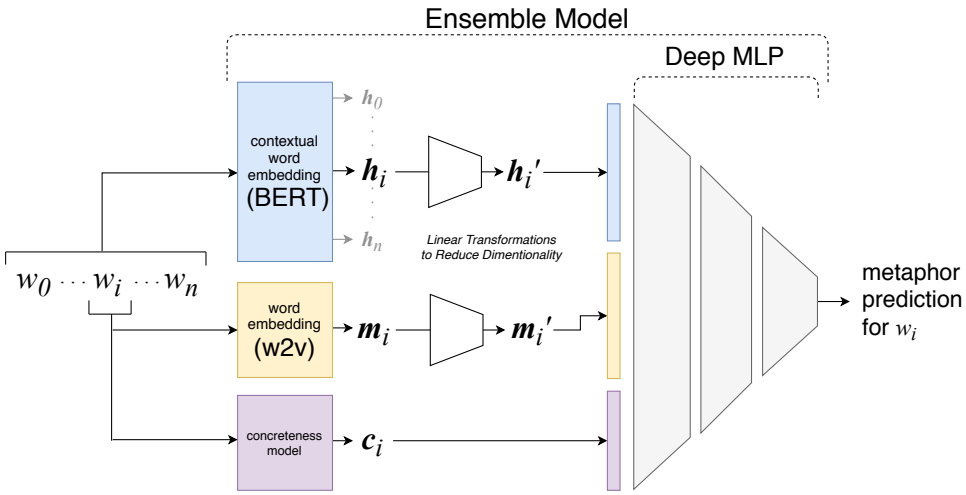

Figure 2: Architecture of our ensemble

concreteness. The output vector of the model is defined as

$$\mathbf{c} = \frac{c}{5} \cdot \mathbf{v}_{\text{con}} + \frac{5-c}{5} \cdot \mathbf{v}_{\text{abs}} \quad (1)$$

Out-of-vocabulary words have their own vector, $\mathbf{v}_{\text{unk}}$. Each of $\mathbf{v}_{\text{abs}}$, $\mathbf{v}_{\text{con}}$, and $\mathbf{v}_{\text{unk}}$ are randomly initialised and learned from data. The dimensionality of these vectors is a hyperparameter which is tuned—a higher dimensionalality will likely place more emphasis on this feature when it is fed into the MLP as part of the ensemble model.

Static Word Embedding We initialise a matrix of static 300-dimensional word embeddings from the *Word2Vec* Google News pretrained model (Mikolov et al., 2013), then fine-tune it to the data. Out-of-vocabulary words are given their lemma's embedding, if present, otherwise they are initialised randomly.

Multi-Layer Perceptron We define a deep multi-layer percepton (MLP), which at each layer has four components: a linear transformation, layer normalisation, a ReLU activation function, and finally dropout (Srivastava et al., 2014). The structure is parameterised with three parameters—the input size k, number of layers n, and first hidden layer size h. The first linear layer of the network has an input of size k, and an output of size h. Each successive layer halves the size of the hidden state. After n layers, a final linear layer converts to a single output, which is then passed through a sigmoid to yield the prediction. Based on this design, we have the constraint that $2^n \leq h \leq k$, which imposes that (1) the first hidden layer is not larger than the input, and (2) the size of the hidden layer does not reach 1 before the final layer.

Ensemble Model Tying all of the aforementioned models together is an ensemble model. First, it passes each input sentence $w_0, \cdots, w_n$ through the contextual embedding component to yield the embeddings $\mathbf{h}_1, \cdots, \mathbf{h}_n$. To reduce their dimentionality, these are each passed through a simple linear transform, yielding $\mathbf{h}'_1, \cdots, \mathbf{h}'_n$. Each word is then passed through the static embedding component, yielding embeddings $\mathbf{m}_1, \cdots, \mathbf{m}_n$, which are also projected down to $\mathbf{m}'_1, \cdots, \mathbf{m}'_n$. Each word is also fed to the concreteness model, yielding concreteness vectors $\mathbf{c}_1, \cdots, \mathbf{c}_n$. For each word, the three representations ($\mathbf{h}'_i$, $\mathbf{m}'_i$, and $\mathbf{c}_i$) are concatenated, and passed into a deep multi-layer perceptron which makes the final metaphor prediction (per-word). This model is depicted in Figure 2. Crucially, though this model accepts sentences (needed to process the contextualised word representations), it makes predictions using the MLP on a per-word basis—but back-propagates through BERT for all annotated words in each sentence.

4 Experiments

Data We use the VUA corpus (Steen et al., 2010) which was made available for the shared task (Leong et al., 2020).[3] We train a model on all

[3]This corpus consists of four different genres from the British National Corpus: academic, news, conversation, and fiction. For the Shared Task, these were all merged, but clearly each categories' data will be of radically different forms. We expect our system to underperform on the transcriptions of conversations, since this will be very different from the data BERT was trained on.

Subset	Train	Dev	Total
By Tokens			
Verbs	15,323	1,917	17,240
All-POS	64,537	8,074	72,611
Everything	161,335	20,169	181,504
By Sentences			
Verbs	7,127	746	7,873
All-POS	9,809	1,085	10,894
Everything	10,738	1,371	12,109

Table 1: Number of tokens and sentences in the data

Parameter	Value
n-layers	4
Hidden size (h)	140
Size of $\mathbf{c}_i$	50
Size of $\mathbf{h}'_i$	150
Size of $\mathbf{m}'_i$	200
Learning rate I	2×10^{-4}
Learning rate II	2×10^{-5}
Weight Decay	0.05
Dropout	0.4

Table 2: Hyperparameters of the final model

Ensemble Model			
CWE	*SWE*	*CM*	F_1 Dev
✓			0.574
✓		✓	0.586
✓	✓		0.636
✓	✓	✓	0.644

Table 3: Ablation study results

the available training data—not just those marked for the *verbs* or *all-pos* subtasks, because we found this improved performance. We split the data in an 8:1 ratio, ensuring that the split puts $1/9$ of each subtask's data in the development set—details of the splits are shown in Table 1.

Training Details We train in batches of 32 sentences, and employ early stopping after 20 stable steps (based on F_1 on dev). As an optimizer, we use AdamW (Loshchilov and Hutter, 2017). We experimented with three fine-tuning options: (1) unfreezing the whole network and training it all at once, (2) freezing BERT and training until early stopping activates, then unfreezing BERT and training until early stopping again, and (3) freezing BERT and training until early stopping, then sequentially unfreezing and training a single layer of BERT at a time, and finally the whole model at once (inspired by Felbo et al., 2017). We used option (2) in the end, since it offered a large improvement over (1) when we used a lower learning rate for the second phase. We found that (3) offered no additional advantage. To find hyperparameters, we performed a random search over the parameter space; final hyperparameters are reported in Table 2.

Threshold Shifting The ratio of metaphors to non-metaphors in the entire VUA dataset was not the same as that of the verb and all-pos subsets used by the Shared Task. Having trained the model on all the data, we then adjust it to each different distribution. To do this, we find the threshold for the sigmoid output that maximises the F_1 score on each particular development set.[4]

Ablation Study To verify that each feature contributes useful information over just using a contextualised representation, we first conduct a simple ablation study, to see the performance impact of removing either the static word embeddings or concreteness ratings. We train four models (with the same hyperparameters as in Table 2), with different combinations of the concreteness model (CM) and static word embedding (SWE) model removed.[5] Table 3 shows the results on the development set. The contextualised word embeddings (CWE) on their own performs the worst. Adding the embeddings in particular really bolsters the performance (increasing it from 0.574 to 0.636 F_1). The type-level concreteness annotation also helps, but not quite as much. The combination of all three features achieves the highest F_1 score.

Shared Task Performance The Shared Task results are computed as the F_1 Score on held-out test data. Our results are presented in Table 4, alongside the results from the previous highest-performing system (Wu et al., 2018) from the 2018 Shared Task (Leong et al., 2018), and the highest-performing system on this shared task. Through the use of contextualised representations and concreteness rat-

[4]We also experimented with fine-tuning the network to each subset, but found this led to overfitting, and was detrimental to performance.

[5]For this experiment we use early stopping after 5 stable results (based on loss), BERT base rather than large, and did not fine-tune BERT.

Model	Subtask F_1	
	Verbs	*All-POS*
Us	0.755	0.718
2018 Winner	0.672	0.651
2020 Winner	0.804	0.769

Table 4: Shared Task results

ings, we are able to improve substantially over the best submission to the 2018 shared task (Wu et al., 2018) for metaphor detection on the VUA corpus (Steen et al., 2010), by 8.0% F_1. We trail the winner of the 2020 task by an average of 5.0% F_1.

5 Conclusion

We participated in the 2020 Metaphor Identification Shared Task (Leong et al., 2020). Our model was designed to try and exploit knowledge of lexical concreteness and contextual meaning to identify metaphors. Our results improved over the previous best performing system by an average of 8.0% F_1, but trailed behind the leader of the task by 5.0%.

In future work, we are keen to explore first training a model to identify concreteness in context, then fine-tuning this to metaphor identification, based on the reasoning presented in §2.

References

Beata Beigman Klebanov, Chee Wee Leong, and Michael Flor. 2015. Supervised word-level metaphor detection: Experiments with concreteness and reweighting of examples. In *Proceedings of the Third Workshop on Metaphor in NLP*, pages 11–20, Denver, Colorado. Association for Computational Linguistics.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904–911.

Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology Section A*, 33(4):497–505.

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

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark. 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.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University of Chicago Press.

Chee Wee Leong, Beata Beigman Klebanov, Chris Hamill, Egon Stemle, Rutuja Ubale, and Xianyang Chen. 2020. A report on the 2020 vua and toefl metaphor detection shared task. In *Proceedings of the Second Workshop on Figurative Language Processing*, Seattle, WA.

Chee Wee (Ben) Leong, Beata Beigman Klebanov, and Ekaterina Shutova. 2018. A report on the 2018 VUA metaphor detection shared task. In *Proceedings of the Workshop on Figurative Language Processing*, pages 56–66, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Rui Mao, Chenghua Lin, and Frank Guerin. 2019. End-to-end sequential metaphor identification inspired by linguistic theories. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3888–3898, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Allan Paivio, John C Yuille, and Stephen A Madigan. 1968. Concreteness, imagery, and meaningfulness values for 925 nouns. *Journal of Experimental Psychology*, 76(1, Pt.2):1–25.

Otfried Spreen and Rudolph W. Schulz. 1966. Parameters of abstraction, meaningfulness, and pronunciability for 329 nouns. *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Gerard J. Steen, Aletta G. Dorst, J Berenike Herrmann, Anna A. Kaal, and Tina Krennmayr. 2010. Metaphor in usage. *Cognitive Linguistics*, 21(4):765–796.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 248–258, Baltimore, Maryland. Association for Computational Linguistics.

Yulia Tsvetkov, Elena Mukomel, and Anatole Gershman. 2013. Cross-lingual metaphor detection using common semantic features. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 45–51, Atlanta, Georgia. Association for Computational Linguistics.

Peter Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 680–690, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. Neural metaphor detecting with CNN-LSTM model. In *Proceedings of the Workshop on Figurative Language Processing*, pages 110–114, New Orleans, Louisiana. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.