Character aware models with similarity learning for metaphor detection

Tarun Kumar
Department of CSIS
Birla Institute of Technology
and Science, Pilani, India
f2016005@pilani.bits-pilani.ac.in

Yashvardhan Sharma
Department of CSIS
Birla Institute of Technology
and Science, Pilani, India
yash@pilani.bits-pilani.ac.in

Abstract

Recent work on automatic sequential metaphor detection has involved recurrent neural networks initialized with different pre-trained word embeddings and which are sometimes combined with hand engineered features. To capture lexical and orthographic information automatically, in this paper we propose to add character based word representation. Also, to contrast the difference between literal and contextual meaning, we utilize a similarity network. We explore these components via two different architectures - a BiLSTM model and a Transformer Encoder model similar to BERT to perform metaphor identification. We participate in the Second Shared Task on Metaphor Detection on both the VUA and TOFEL datasets with the above models. The experimental results demonstrate the effectiveness of our method as it outperforms all the systems which participated in the previous shared task.

1 Introduction

Metaphors are an inherent component of natural language and enrich our day-to-day communication both in verbal and written forms. A metaphoric expression involves the use of one domain or concept to explain or represent another concept (Lakoff and Johnson, 1980). Detecting metaphors is a crucial step in interpreting semantic information and thus building better representations for natural language understanding (Shutova and Teufel, 2010). This is beneficial for applications which require to infer the literal/metaphorical usage of words such as information extraction, conversational systems and sentiment analysis (Tsvetkov et al., 2014).

The detection of metaphorical usage is not a trivial task. For example, in phrases such as *breaking the habit* and *absorption of knowledge*, the words *breaking* and *absorption* are used metaphorically to mean to destroy/end and understand/learn respectively. In the phrase, *All the world's a stage*, the *world* (abstract) has been portrayed in a more concrete (*stage*) sense. Thus, computational approaches to metaphor identification need to exploit world knowledge, context and domain understanding (Tsvetkov et al., 2014).

A number of approaches to metaphor detection have been proposed in the last decade. Many of them use explicit hand-engineered lexical and syntactic information (Hovy et al., 2013; Klebanov et al., 2016), higher level features such as concreteness scores (Turney et al., 2011; Köper and Schulte im Walde, 2017) and WordNet supersenses (Tsvetkov et al., 2014). The more recent methods have modeled metaphor detection as a sequence labeling task, and hence have used BiLSTM (Graves and Schmidhuber, 2005) in different ways (Wu et al., 2018; Gao et al., 2018; Mao et al., 2019; Bizzoni and Ghanimifard, 2018).

In this paper, we use concatenation of GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) vectors augmented with character level features using CNN and highway network (Kim et al., 2016; Srivastava et al., 2015). Such a method of combining pre-trained embeddings with character level representations has been previously used in several sequence tagging tasks - part-of-speech (POS) tagging (Ma and Hovy, 2016) and named entity recognition (NER) (Chiu and Nichols, 2016), question answering (Seo et al., 2016) and multitask learning (Sanh et al., 2019). This inspires us to explore similar setting for metaphor identification as well.

We propose two models for metaphor detection[1] with the input prepared as above - a vanilla BiLSTM model and a vanilla Transformer Encoder (Vaswani et al., 2017) model similar to BERT (Devlin et al., 2019) (but without pre-training). To contrast the difference between a word's literal and contextual representation (Mao et al., 2019) con-

[1]Our code is available at: https://github.com/Kumar-Tarun/metaphor-detection

Proceedings of the Second Workshop on Figurative Language Processing, pages 116–125
July 9, 2020. ©2020 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17

catenated the two before feeding into the softmax classifier. Instead, we extend the idea of cosine similarity between two words in a phrase of signifying metaphoricity (Shutova et al., 2016; Rei et al., 2017) to similarity between the literal and contextual representations of a word and then feed this result into the classifier.

Finally, we participate in The Second Shared Task on Metaphor Detection[2] on both the VU Amsterdam Metaphor Corpus (VUA) (Steen et al., 2010) and TOEFL, a subset of ETS Corpus of Non-Native Written English (Beigman Klebanov et al., 2018) datasets with the above models and a vanilla combination of them. The combination of the models outperforms the winner (Wu et al., 2018) of the previous shared task (Leong et al., 2018).

2 Related Work

Previous metaphor detection frameworks include supervised machine learning approaches utilizing explicit hand-engineered features, approaches based on unsupervised learning and representation learning, and deep learning models to detect metaphors in an end-to-end manner. (Köper and Schulte im Walde, 2017) determine the difference of concreteness scores between the target word and its context and use this to predict the metaphoricity of verbs in the VUA dataset. (Tsvetkov et al., 2014) combine vector space representations with features such as abstractness and imageability and WordNet Supersenses to model the metaphor detection problem in two syntactic constructions - subject-verb-object (SVO) and adjective-noun (AN). Evaluating their approach on the TroFi dataset (Birke and Sarkar, 2006), they achieve competitive accuracy. (Hovy et al., 2013) explore differences in compositional behaviour of a word's literal and metaphorical use in certain syntactic settings. Using lexical, WordNet supersense features and PoS tags of sentence tree, they train an SVM using tree-kernel. (Klebanov et al., 2016) use semantic classes of verbs such as orthographic unigram, lemma unigram, distributional clusters etc. to identify metaphors in the VUA dataset.

Some of the methods for metaphor detection utilize unsupervised learning. (Mao et al., 2018) train word embeddings on wikipedia dump and use WordNet compute a best-fit word corresponding to a target word in a sentence. The cosine similarity between these two words indicates the metaphoricity of the target word. (Shutova et al., 2016) compute word embeddings and phrase embeddings on wikipedia dump. They extract visual features from CNNs using images from Google Images. Next, multimodal fusion strategies are explored to determine metaphoricity.

Recently, approaches based on deep learning have been proposed. The first in this line is Supervised Similarity network by (Rei et al., 2017). They capture metaphoric composition by modeling the interaction between source and target domain by a gating function and then using a cosine similarity network to compute metaphoricity. They evaluate their method on adjective-noun, verb-subject and verb-direct object constructions on the MOH (Mohammad et al., 2016) and TSV (Tsvetkov et al., 2014) datasets.

More recently, the problem has been modeled as a sequence labeling task, in which at each timestep the word is predicted as literal or metaphoric. (Wu et al., 2018) used word2vec (Mikolov et al., 2013), PoS tags and word clusters as input features to a CNN and BiLSTM network. They compared inference using softmax and CRF layers, and found softmax to work better. (Bizzoni and Ghanimifard, 2018) propose two models - a BiLSTM with dense layers before and after it and a recursive model for bigram phrase composition using fully-connected neural network. They also added concreteness scores to boost performance. (Gao et al., 2018) fed GloVe and ELMo embeddings into a vanilla BiLSTM followed by softmax. (Mao et al., 2019) proposed models based on MIP (Group, 2007) and SVP (Wilks, 1975, 1978) linguistic theories and achieved competitive performance on VUA, MOH and TroFi datasets.

3 Methodology

In this paper we propose two architectures for metaphor detection based on sequence labeling paradigm - a BiLSTM model and a Transformer Encoder model. Both the models are initialized with rich word representations. First, we describe the word representations, then, the similarity network, and subsequently the models (Figure 1).

3.1 Word Representations

The first step in building word representations is the concatenation of GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings. The

[2]`https://competitions.codalab.org/competitions/22188`

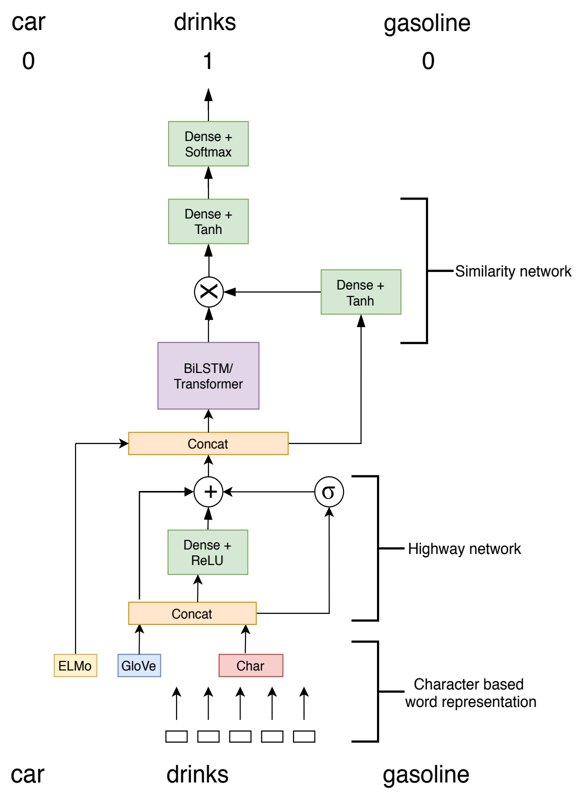

Figure 1: Proposed model which includes character embeddings and similarity network

combination of these two have shown good performance across an array of NLP tasks (Peters et al., 2018). While these two representations are based on corpus statistics and bidirectional language models respectively and serve as a good starting point as shown by (Gao et al., 2018) and (Mao et al., 2019), however to learn explicit lexical, syntactic and orthographic information (so as to be more suited for metaphor tasks) we augment these word representations with character level embeddings. We follow (Kim et al., 2016) to compute character-level representations by a 1D CNN (see Figure 2) followed by a highway network (Srivastava et al., 2015).

Let word at position t be made up of characters $[c_1, \ldots, c_l]$, where each $c_i \in R^d$, l is the length of word and d is dimensionality[3] of character embeddings. Let $C^t \in \mathbb{R}^{d \times l}$ denote the character-level embedding matrix of word t. This matrix is convolved with filter $H \in \mathbb{R}^{d \times w}$ of width w, followed by a non-linearity.

$$f^t = \tanh(C^t * H + b), f^t \in \mathbb{R}^{l-w+1} \tag{1}$$

Next, we apply max-pooling over the length of f to get a output for one filter.

$$y^t = \max_{1 \le j \le l-w+1} \{f_j^t\} \tag{2}$$

Now, we take multiple filters of different widths and concatenate the output of each to get a vector representation of word t. Let h be the number of filters and $y_1, \ldots, y_h$ be the outputs, then $c_t = [y_1^t, \ldots, y_h^t]$. We concatenate GloVe embedding (g_t) with c_t and run it through a single layer highway network (Srivastava et al., 2015).

$$a_t = [g_t; c_t] \tag{3}$$
$$t = \sigma(W_T a_t + b_T) \tag{4}$$
$$z_t = t \odot g(W_H a_t + b_H) + (1 - t) \odot a_t \tag{5}$$

z_t and a_t have same dimensionality by construction, W_H and W_T are square matrices, g is ReLU activation. t is called as transform gate and $(1 - t)$ as the carry gate. The role of highway network is to select the dimensions which are to be modified and which are to be passed directly to output. Thus, we allow the network to adjust the contribution of GloVe and character-based embeddings for better learning (thus an adjustment between semantic and lexical information). We also concatenated GloVe, ELMo and character embeddings and passed through highway layer, but the former approach performed better with lesser parameters. Our input representation is $[z_t; e_t]$ (where e_t is ELMo vector) which is fed to BiLSTM/Transformer.

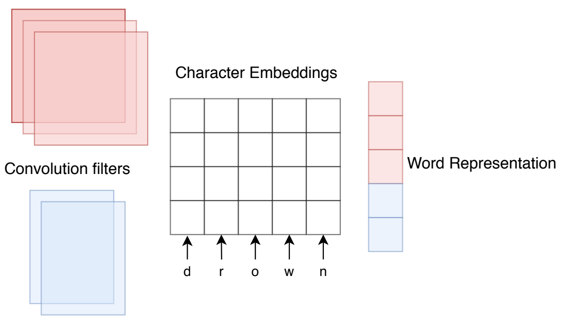

Figure 2: CNN for extracting character-level representations

3.2 BiLSTM model

We use a single-layer BiLSTM model (Graves and Schmidhuber, 2005) to produce hidden states h_t for each position t. These hidden states represent our contextual meaning, the meaning which we will contrast with the input literal meaning. Using

[3] d is chosen less than the $|C|$, the size of vocabulary of characters

hidden states as a candidate for contextual meaning has been done previously (Gao et al., 2018; Mao et al., 2019; Wu et al., 2018). A simple approach would be to pass h_t directly to softmax layer for predictions. But we condition our predictions both on h_t and input representation as shown in next sub-section.

3.3 Similarity Network

(Rei et al., 2017) use a weighted cosine similarity network to determine similarity between two word vectors in a phrase (Shutova et al., 2016). We extend this idea further to calculation of similarity between literal and contextual representations. To perform this computation, we first project the input embeddings to the size of hidden dimension of BiLSTM.

$$x_t = [z_t; e_t] \tag{6}$$

$$\tilde{x_t} = \tanh(W_z x_t) \tag{7}$$

This step serves two purposes - first reduces the size to enable calculation, second performs vector space mapping. Since input embeddings are in a different semantic vector space (due to the pre-trained vectors), we allow the network to learn a mapping to the more metaphor specific vector space. Next, we element-wise multiply $\tilde{x_t}$ with h_t.

$$m_t = \tilde{x_t} \odot h_t \tag{8}$$

m_t is input to a dense layer as follows,

$$u_t = \tanh(W_u m_t) \tag{9}$$

If u_t has length 1, W_u has all weights equal to 1 and linear activation is used instead of $\tanh$, then the above two steps mimic the cosine similarity function. But, to provide better generalization, $|u_t| > 1$ and $\tanh$ is used to allow the model to learn custom features for metaphor detection (Rei et al., 2017). u_t is fed to softmax classifier to make predictions.

$$p(\hat{y_t}|u_t) = \sigma(W_y u_t + b) \tag{10}$$

σ is the softmax function, W_y and b are trainable weights and bias respectively.

3.4 Transformer model

The advent of Transformer (Vaswani et al., 2017) and further general language models such as BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019) have shown excellent performance across multiple NLP, NLU and NLG tasks. Inspired by this, we explore a vanilla transformer model in this paper which consists of only the encoder stack and is not pre-trained on any corpus.

The input to the transformer model is the same as the BiLSTM model. To contrast the literal meaning with the contextual meaning, we employ equations 6,7,8,9,10 except that h_t would denote the output of the transformer at position t. (Mao et al., 2019) also explored transformers in their experiments, but they only computed word representations from a pre-trained BERT large model and fed it to BiLSTM, they did not train a transformer model from scratch. Since transformers do not track positional information, positional encodings are added for this purpose, but in our case adding such encoding did not improve performance. Furthermore, our transformer model is composed of only a single transformer block (that is depth=1) with a single head. Such a simple model is able to reach good score on the metaphor detection task.

Dataset	Train			Test	
	#T	#S	%M	#T	#S
VUA All-POS	72,611	12,122	15%	22,196	4,080
VUA Verb	17,240	12,122	28%	5,873	4,080
TOEFL All-POS	26,737	2741	7%	9,017	968
TOEFL Verb	7,016	2741	14%	2,301	968

Table 1: Dataset Statistics. #T denotes the number of tokens which are annotated. #S denotes the number of sentences. %M denotes the token-level metaphoricity percentage.

4 Experiment

4.1 Dataset

We evaluate our models on two metaphor datasets on both ALL-POS and VERB track in the Second Shared Task on Metaphor Detection. Table 1 shows the dataset statistics.

First is the VU Amsterdam Metaphor Corpus (VUA) (Steen et al., 2010) widely studied dataset for metaphor detection. All the words in this dataset are labeled as either metaphoric or literal according to MIPVU (Steen et al., 2010; Group, 2007) protocol. This dataset was also used in the 2018 Shared Task on Metaphor Detection (Leong et al., 2018).

Second is the TOEFL corpus, a subset of ETS Corpus of Non-Native Written English

(Beigman Klebanov et al., 2018). This dataset contains the essays written by takers of the TOEFL test having either medium or high English proficiency. The words in this dataset are annotated for argumentation-relevant metaphors. The essays are in response to prompts, for which test-takers were required to argue for or against and in such process the metaphors used to support one's argument were annotated. So, the protocol used (Beigman Klebanov and Flor, 2013) is different from MIPVU.

4.2 Baselines

The first four baselines are evaluated on the VUA test set and the last two on the TOEFL test set. **CNN-BiLSTM** (Wu et al., 2018): This model is the winner of the previous shared task (Leong et al., 2018). They proposed an ensemble of CNN-BiLSTM network with input features as word2vec, PoS tags and word2vec clusters.
BiLSTM (Gao et al., 2018) : This model is a simple BiLSTM with inputs as concatenation of GloVe and ELMo embeddings.
BiLSTM-MCHA (Mao et al., 2019) : This model employs BiLSTM followed by a multi-head contextual attention which is inspired by SPV protocol of metaphor identification. They also use GloVe and ELMo as input features.
BiLSTM-Concat (Bizzoni and Ghanimifard, 2018) : This model achieved the second position in the previous shared task. They combined a BiLSTM (preceded and followed by dense layers) and a model based on recursive composition of word embedding. Concreteness scores were added to boost performance.
CE-BiLSTM : We add a variant of our proposed model without the Transformer model and the similarity network. All other components are kept same. CE denotes character embeddings.
Feature-based (Beigman Klebanov et al., 2018) : They use several hand-crafted features and train a logistic regression classifier to predict metaphoricity. This is the only known work on TOEFL dataset to the best of our knowledge.

We note that BiLSTM and BiLSTM-MHCA models above have different experimental settings than ours. They trained and tested their models on different amount of data when compared to the shared task. For a fair comparison, we evaluate (train and test) our method in the same data setting (Table 3).

4.3 Setup

The 300d pre-trained GloVe embeddings are used along with 1024d pre-trained ELMo embeddings. The dimension of character-level embeddings is set to 50. The filters used in CharCNN are $[(1, 25), (2, 50), (3, 75), (4, 100)]$, where first element of each tuple denotes the width of filter and second element denotes the number of filters used. Inspired by the effectiveness of PoS tags (Wu et al., 2018; Beigman Klebanov et al., 2014) in metaphor detection, we concatenate 30 dimensional PoS embeddings. We found 30d embeddings to work better than one-hot encodings. These embeddings are learned during model training. The uni-directional hidden state size of BiLSTM is set to 300. We apply Dropout (Srivastava et al., 2014) on input to BiLSTM and to the output of BiLSTM. The dimension of u_t, the output size of similarity network is set to 50.

The hidden state size of Transformer is set to 300 as well. We use a single head and single layer architecture. We also tried multiple heads (8, 16), but the performance dropped a little. The attention due to padded tokens is masked out in the attention matrix during forward pass. The feed-forward network which is applied after the self-attention layer consists of two linear transformations with ReLU activation in between (Vaswani et al., 2017). First transformation projects 300d to 1200d and second transformation projects 1200d back to 300d. Dropout is applied both before and after the feed-forward network. It can be seen that this transformer model is simplified in terms of number of parameters when compared to BERT (Devlin et al., 2019). Our focus here is on the power of transformer architecture rather than on transformer based huge language models.

We also explore the combination of both the models. Specifically, BiLSTM and Transformer model are combined at the pre-activation stage, that is, the logits of both networks are averaged and then input to the softmax layer for predictions. Both the models are trained in parallel, with their own losses, whereas the F1-score is calculated from the combined prediction.

The objective function used is weighted cross-entropy loss as used in (Mao et al., 2019; Wu et al., 2018).

$$L = -\sum_{n=1}^{M} w_{y_n} y_n \log(\hat{y_n}) \tag{11}$$

Model	VUA ALL POS			VUA VERB		
	P	R	F1	P	R	F1
CNN-BiLSTM	60.8	70.0	65.1	60.0	76.3	67.1
BiLSTM-Concat	59.5	68.0	63.5	-	-	-
CE-BiLSTM-Transformer	60.6	73.9	66.6	62.7	82.2	71.2
CE-BiLSTM-Transformer (Ensemble)	63.0	71.6	**67.0**	66.7	77.5	**71.7**

Table 2: Comparison of our method against the baseline systems on the VUA test set.

Model	VUA ALL POS			VUA VERB		
	P	R	F1	P	R	F1
BiLSTM	71.6	73.6	72.6	68.2	71.3	69.7
BiLSTM-MHCA	73.0	75.7	74.3	66.3	75.2	70.5
CE-BiLSTM-Transformer	71.3	78.5	74.7	66.1	76.2	70.8
CE-BiLSTM-Transformer (Ensemble)	75.9	74.1	**75.0**	68.0	75.1	**71.4**

Table 3: Comparison of our method against the baseline systems on the VUA test set with different experimental setting.

Model	TOEFL ALL POS			TOEFL VERB		
	P	R	F1	P	R	F1
Feature-based	49.0	58.0	53.0	50.0	64.0	56.0
CE-BiLSTM	62.7	60.8	61.8	70.0	60.5	64.9
CE-BiLSTM-Transformer	62.3	61.7	**62.0**	66.9	63.8	**65.3**

Table 4: Comparison of our method against the baseline systems on the TOEFL test set.

where y_n is the gold label, $\hat{y_n}$ is the predicted score and w_{y_n} is set to 1 if y_n is literal and 2 otherwise. We use Adam optimizer (Kingma and Ba, 2014) and early stopping on the basis of validation F-score. Batch-size is set to 4.

TOEFL dataset contains essays annotated for metaphor and metadata mapping essays to the respective prompts and English proficiency of test-takers. We extract all sentences from all the essays and prepare our dataset considering one sentence as one example (batch-size x means x such examples). In this paper, we do not exploit the metadata of TOEFL corpus.

For both VUA and TOEFL datasets, we have a pre-specified train and test partition, so for hyper-parameter tuning we split the train set into train and validation in the ratio of 10:1 randomly. Since the models predict labels for all the words in a sequence, we train a single model and use it for evaluating both ALL-POS and Verb tracks. We report F-score on test set for metaphor class on both datasets and tasks. Section 6 presents an ablation study and explores the performance of different components.

5 Results

We first compare our method against the baseline systems which have the same experimental setting as ours on the VUA test set - CNN-BiLSTM and BiLSTM-Concat. Table 2 reports the results. As shown, our proposed model (comprising of both BiLSTM and Transformer) outperforms the other methods on both the tracks. Specifically, we achieve F-score of 66.6 on VUA All POS and 71.2 for VUA Verb set. Furthermore, we employ ensembling to boost our performance. This strategy mainly improves precision (60.6 to 63.0 for All POS, 62.7 to 66.7 for Verb). For ensembling we run the model 7 times which involves different dropout probabilities, changing the ratio of metaphoric to literal loss weights, increasing/decreasing number of epochs. Thus, we do not modify the number of parameters in any run. At the end, we take a majority vote to produce final predictions. Our best F-score on All POS track is 67.0 and Verb track is 71.7. We observe higher F-scores on Verb track than on All POS track, this might be due to fact that a higher percentage of verbs are annotated as being metaphoric, hence more training data.

We now compare our method with the other two baselines on a common experimental setting. We tune our hyperparameters in this setting due to difference in training and validation data. Specifically, since training set is of smaller size, we increase Dropout probabilities, and the dimension of PoS embedding is reduced from 30 to 10. As shown in Table 3, the single best model achieves a higher F-score than the baselines and the ensemble (with similar setting as above) improves the performance

Model	VUA Validation		
	P	R	F1
Vanilla BiLSTM	61.7	82.6	70.6
Vanilla Transformer	63.6	75.6	69.1
Vanilla BiLSTM + Transformer	67.1	77.7	72.0

Table 5: Performance of vanilla models on VUA validation set.

Model	VUA Validation		
	P	R	F1
CE + BiLSTM	65.7	78.8	71.6
CE + Transformer	67.3	74.7	70.8
CE + BiLSTM + Transformer	71.3	74.2	72.7

Table 6: Addition of Character embeddings to models.

a little more scoring 75.0 on All POS and 71.4 on Verb tracks.

Lastly, we explore the performance of our method on the TOEFL test set (Table 4). We added an extra baseline which does not include the Transformer model and the similarity network. Also, the CE-BiLSTM-Transformer model here does not include the similarity network. The reason for this is because it degraded performance. The similarity network contrasts the literal meaning with the contextual meaning of the target word which is in line with MIP (Steen et al., 2010) protocol. Since, TOEFL corpus is annotated for argument-specific metaphors and not MIP, we hypothesize that this might be the reason for lower performance. However, VUA is annotated according to MIP, thus similarity component improves performance here, as we show in the ablation section.

Table 4 shows that both our baseline (CE-BiLSTM) and baseline + Transformer improve upon the Feature-based model by 8.8 and 9.0 points respectively on All POS track and 8.9 and 9.3 points respectively on Verb track. Similar to VUA, here also Verbs score higher than All POS because of more training instances for verbs.

The scores on TOEFL dataset are lower than the VUA dataset. This is due to the lesser number of training instances in TOEFL dataset. Also, while we have higher recall on VUA, on TOEFL we have higher precision.

6 Ablation Study

This section considers the performance of different components of our method in isolation and combination on the VUA validation set unless otherwise specified. The reason for choosing validation set is because we were not able to evaluate some settings on the test set due to limited time and number of submissions. Wherever we have test set results we report those as well.

Impact of Character Embeddings We first note the performances of vanilla BiLSTM and vanilla Transformer models and a simple combination of them in Table 5. Note that vanilla implementation still includes GloVe and ELMo vectors. We see that BiLSTM performs better than Transformer model and that a combination of them seems to complement each other.

Now, we see the impact of adding character-level embeddings on both the models. As Table 6 shows, addition of character embeddings improves both the networks. Particularly, Transformer benefits more from this addition as F1-score increases from 69.1 to 70.8. On the test set, our vanilla combination scores 65.2 whereas the combination of models with character embeddings scores 66.1. This helps in asserting the usefulness of character-based features in learning pro-metaphor features. (Beigman Klebanov et al., 2016) demonstrate the utility of unigram lemmas and orthographic features in metaphor detection. Our character embeddings computed from CNN combines features at different n-grams of a word and thus helps to learn lexical and orthographic information automatically which aids in improving performance.

We suspect that employing the baseline unigram features (Beigman Klebanov et al., 2014) provided by the organizers instead of learned character-embeddings may be seen as a way to achieve the same goal. But our method is more robust in the sense that, we allow for learning of different n-gram features of a word (including unigram itself). Particularly, our method is helpful in cases where the target word has incorrect spelling, because we learn representations instead of using fixed pre-computed features.

Impact of Similarity Network Table 7 depicts the performance after the addition of similarity network. As the similarity network is guided by the MIP protocol, it indeed boosts results for the VUA dataset. We observe that in this case too Transformer benefits more by the inclusion and the benefit (1.9 points) is even more than by adding character embeddings (1.7 points). However, for both the components increments in BiLSTM performance are equal. Also, the combination of both models with similarity network outperforms the combination with character embeddings although

Model	VUA Validation		
	P	R	F1
SN + BiLSTM	66.1	78.1	71.6
SN + Transformer	70.2	74.0	72.0
SN + BiLSTM + Transformer	68.7	77.8	73.0

Table 7: Addition of Similarity network to models. (SN is the Similarity network)

Model	VUA Validation		
	P	R	F1
CE + SN + BiLSTM	66.7	79.3	72.4
CE + SN + Transformer	67.7	77.8	72.4
CE + SN + BiLSTM + Transformer	68.5	79.1	73.4

Table 8: Addition of both SN and CE to the models.

by a small margin. The above reasoning indicates towards similarity network as being an important component for detection of MIP guided labeling of metaphors.

Table 8 reports the numbers when both character embeddings and similarity network are added to the base models. The results improve from either of the additions which indicate that they complement each other. Our best model so far contains both the base models and the components. This model on the VUA test set, scores 66.5 and the model in the last row of Table 6 scores 66.1.

In all the cases examined till now, Transformer based models have higher precision than the BiLSTM based models, and in 3 out of 4 cases of (Vanilla, CE, SN, CE + SN), the combination has as even better precision than either of the individual models. In terms of F-score, BiLSTM based models score higher than Transformer based ones in 2 cases (Vanilla and CE), equal in CE + SN and lower in SN.

Impact of PoS tags Incorporation of PoS tags proves to be beneficial. It improves the F-score of the last model in Table 8 from 73.4 to 73.5. On the test set, it improves the F-score from 66.5 to 66.6 which is in line with (Hovy et al., 2013; Wu et al., 2018).

7 Conclusion

We proposed two metaphor detection models, a BiLSTM model based on prior work and a Transformer model based on their success in NLP tasks. We augment these models with two components - Character Embeddings and Similarity network to learn lexical features and contrast literal and contextual meanings respectively. Our experimental results demonstrate the effectiveness of our method as we achieve superior performance than all the previous methods on VUA corpus and TOEFL corpus. Through an ablation study we examine the contribution of different parts of our framework in the task of metaphor detection.

In our future work we would explore metaphor detection in a multi-task setting with semantically similar tasks such as Word Sense Disambiguation and Co-reference Resolution. These auxiliary tasks may help to better understand the contextual meaning and reach of a word. For TOEFL dataset, future avenues would include strategies to exploit the metadata, and similarity measures more suitable for argumentation-relevant metaphors.

References

Beata Beigman Klebanov and Michael Flor. 2013. Argumentation-relevant metaphors in test-taker essays. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 11–20, Atlanta, Georgia. Association for Computational Linguistics.

Beata Beigman Klebanov, Ben Leong, Michael Heilman, and Michael Flor. 2014. Different texts, same metaphors: Unigrams and beyond. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 11–17, Baltimore, MD. Association for Computational Linguistics.

Beata Beigman Klebanov, Chee Wee Leong, E. Dario Gutierrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 101–106, Berlin, Germany. Association for Computational Linguistics.

Beata Beigman Klebanov, Chee Wee (Ben) Leong, and Michael Flor. 2018. A corpus of non-native written English annotated for metaphor. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 86–91, New Orleans, Louisiana. Association for Computational Linguistics.

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*, Trento, Italy. Association for Computational Linguistics.

Yuri Bizzoni and Mehdi Ghanimifard. 2018. Bigrams and BiLSTMs two neural networks for sequential metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing*, pages 91–101, New Orleans, Louisiana. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

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.

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.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Pragglejaz Group. 2007. Mip: A method for identifying metaphorically used words in discourse. *Metaphor and Symbol*, 22(1):1–39.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Beata Beigman Klebanov, Chee Wee Leong, E Dario Gutierrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 101–106.

Maximilian Köper and Sabine Schulte im Walde. 2017. Improving verb metaphor detection by propagating abstractness to words, phrases and individual senses. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 24–30, Valencia, Spain. Association for Computational Linguistics.

George Lakoff and Mark Johnson. 1980. Metaphors we live by. *Chicago, IL: University of Chicago*.

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.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and WordNet based metaphor identification and interpretation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1222–1231, Melbourne, Australia. Association for Computational Linguistics.

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, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew 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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1537–1546, Copenhagen, Denmark. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

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.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor corpus annotated for source-target domain mappings. In *LREC*, volume 2. Citeseer.

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.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385.

Gerard Steen, Aletta Dorst, J Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

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.

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 *Advances in neural information processing systems*, pages 5998–6008.

Yorick Wilks. 1975. A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, 6(1):53–74.

Yorick Wilks. 1978. Making preferences more active. *Artificial intelligence*, 11(3):197–223.

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.