An Investigation into the Contribution of Locally Aggregated Descriptors to Figurative Language Identification

Sina Mahdipour Saravani[♠] Ritwik Banerjee[♣] Indrakshi Ray[♠]
[♠]Department of Computer Science, Colorado State University
{sinamps, indrakshi.ray}@colostate.edu
[♣]Department of Computer Science, Stony Brook University
rbanerjee@cs.stonybrook.edu

Abstract

In natural language understanding, topics that touch upon figurative language and pragmatics are notably difficult. We probe a novel use of locally aggregated descriptors – specifically, an architecture called NeXtVLAD – motivated by its accomplishments in computer vision, achieve tremendous success in the FigLang2020 sarcasm detection task. The reported F_1 score of 93.1% is 14% higher than the next best result. We specifically investigate the extent to which the novel architecture is responsible for this boost, and find that it does not provide statistically significant benefits. Deep learning approaches are expensive, and we hope our insights highlighting the lack of benefits from introducing a resource-intensive component will aid future research to distill the effective elements from long and complex pipelines, thereby providing a boost to the wider research community.

1 Introduction

Natural language understanding often goes beyond the syntactic and semantic layers, and perhaps nowhere is this more palpable than in the use of figurative language. A better understanding of figurative language use, such as metaphors, irony, or sarcasm, can not only lead to advances in computational creativity (Veale, 2011; Kuznetsova et al., 2013), but also in understanding social media content, where users often employ such pragmatic tools as irony or sarcasm (Reyes et al., 2013; Riloff et al., 2013). This type of figurative language is difficult to identify, however, at least partly due to what the influential literary poet and critic William Empson called "ambiguities" (Empson, 1947) in the language. In particular, figurative language use with sarcasm or irony completely decouples – and even contrasts – the communicator's intent from the communicated content (Camp, 2012), rendering shallow syntactic or semantic features unsuitable. The poor fit of such features is further exacerbated in social media posts due to the ubiquity of grammatical errors, hashtags, emojis, etc.

The deeper, context-dependent inferential nature of figurative language, together with the poor fit of shallow syntactic and semantic features, makes deep neural networks a natural candidate for downstream NLP tasks like sarcasm detection (Ghosh and Veale, 2016). Unfortunately, with increasing popularity of deep learning, the reliability of findings in publications that extensively employ deep learning can be expected, in general, to decrease (Pfeiffer and Hoffmann, 2009). In light of this seminal empirical observation and the general difficulty of accurately identifying figurative language, it is reasonable to not expect outright success on a benchmark corpus simply based on the use of a deep network.

The concerns about reliability, and thus, about reproducibility, are particularly acute in deep learning. For instance, Reimers and Gurevych (2017) demonstrated that the hyperparameter settings have a significant impact on the final results obtained by a model. Crane (2018) further showed that other confounding factors such as variation of GPUs, the exact version of a framework, the randomness of a seed value provided to a learning algorithm, and the interaction between multiple such factors, can all impact the obtained results.

Beyond reproducibility, however, lies another pertinent factor: the use of increasingly complex pipelines where multiple sophisticated components are glued together for an important downstream NLP task. In such scenarios, it is not always clear *which* components within the complex system may be responsible for improved outcomes. A simple change in data preprocessing may lead to a significant difference in the final result, for example (Etaiwi and Naymat, 2017; Camacho-Collados and Pilehvar, 2018). In publications that introduce complex NLP pipelines, however, such details have sometimes been omitted.

Turn	Tweet	Label
Context-1	The [govt] just confiscated a $180 million boat shipment of cocaine from drug traffickers.	Sarcastic
Context-2	People think 5 tonnes is not a load of cocaine.	
Response	Man! I've seen more than that on a Friday night.	

Table 1: A Tweet thread in the FigLang corpus. Sarcasm being context-dependent, the entire thread serves as a single sample. The label is based on the final response in the thread.

Variable	Dataset	Mean	Median	Std
Tweet length (num. tokens)	Train	140.00	128.00	51.57
	Validation	137.00	125.00	51.17
	Test	143.00	138.00	48.56
Thread length (num. tweets)	Train	4.85	4.00	3.20
	Validation	4.93	4.00	3.29
	Test	4.16	3.00	1.95

Table 2: Overview of the FigLang corpus, showing the overall statistics for the size of individual Tweets (using the BERT tokenizer) and the size of Tweet threads.

2 Exemplar task and data

Within the limited scope of this paper, our goal is to specifically investigate the state-of-the-art sarcasm detection system presented by Lee et al. (2020) – which reported an F_1 score of 93.1%, 14% higher than the next best result reported to the FigLang 2020 workshop (Ghosh et al., 2020) for the Twitter track – and to distill a novel deep learning component used in their pipeline in order to investigate its contribution to the final result. Through a comprehensive series of experiments, we find that this novel architecture (discussed in Sections 3 and 4) does not lead to any significant improvement. The improvement may thus be attributed to components other than deep learning, such as augmenting the corpus by using additional data. Investigating the other components, however, is not in the scope of the work being presented here.

The task is to determine if the final response in a thread (*i.e.*, a sequence of Tweets where each post is in response to its previous post) is sarcastic. One such thread is shown in Table 1. All our experiments are conducted on the Twitter corpus of the FigLang 2020 sarcasm detection task (Ghosh et al., 2020), which comprises $5,000$ threads in the training set and $1,800$ in the test set. Additional properties of this corpus are shown in Table 2.

3 Background

The architecture we investigate has recently been used in downstream NLP tasks, motivated by its success in computer vision. Its origins, however, can be traced back to NLP research, when Sivic and Zisserman (2003) borrowed from the bag-of-words approach used in text retrieval. Since then, a significant body of work in computer vision has developed this approach further. The core idea being the treatment of an image as a document, and low-dimensional features[1] extracted from them forming the visual vocabulary, thus enabling a vector representation of each image, subsequently used in classification or ranking tasks.

A key advancement came in the form of Vector of Locally Aggregated Descriptors (VLAD), introduced by Jégou et al. (2010). In this work, too, low-dimensional features were extracted from images, but K clusters of the features were created, and only the difference of each feature from the cluster center was recorded. Instead of a single N-dimensional feature vector, each image would thus be represented by a $K \times N$ matrix.

The non-differentiable hard cluster assignment, however, renders it unsuitable for training a neural network. NetVLAD (Arandjelovic et al., 2016) resolves this by using the softmax function, whose parameters can be learned during training. Since the cluster assignments of a feature are not known prior to training, their approach requires K N-dimensional difference vectors to encode each feature. This increase in the number of parameters impedes model optimization, and may lead to overfitting – drawbacks discussed and subsequently addressed by NeXtVLAD (Lin et al., 2018) by introducing a step prior to the soft cluster assignments. In this step, the input is expanded to λN size by a fully-connected layer, and then decomposed into G groups of lower-dimensional vectors. Further, a sigmoid function with range $[0, 1]$ is used to assign attention scores to the groups for each vector. The process effectively provides a $\frac{G}{\lambda}$ reduction in the number of parameters, by aggregating lower-dimensional vectors. From a linear algebra perspective, this can be interpreted as representing the data using subspace projections of the original vector.

[1]The literature on image processing often uses the term "descriptor", but to stay in tune with the terminology in NLP research, we continue to use the term "feature".

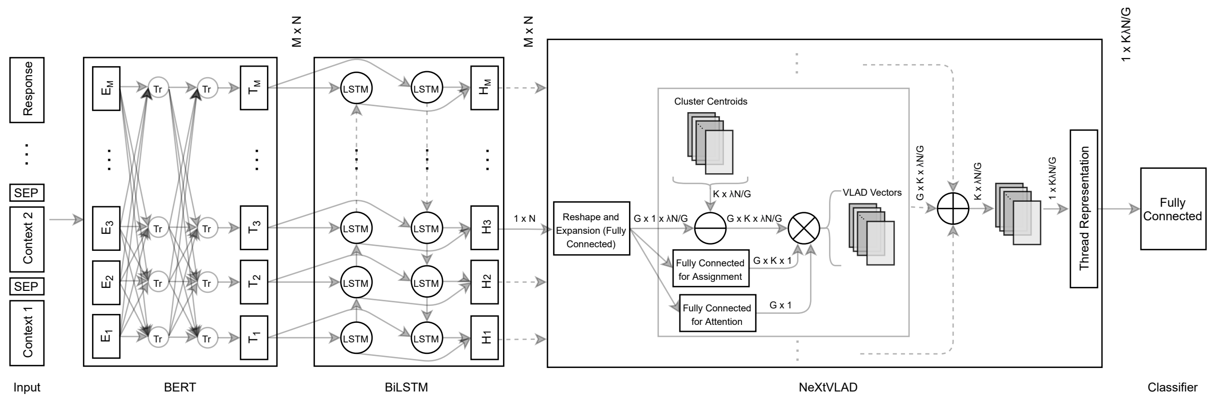

Figure 1: The architecture for sarcasm detection, where M is the number of tokens from the input text, N is the dimension of the BERT representation, and G is the number of groups into which the input is split after expansion.

4 Architecture for sarcasm detection

For an analogous use of NeXtVLAD in NLP, the token representation vectors take the place of the feature vectors used in computer vision literature. In particular, for sarcasm detection using the FigLang corpus, one entire thread needs to be represented by a $K \times N$ matrix. To achieve this, the context and response Tweets (as shown in Table 1) from a single thread are concatenated, with a special [SEP] token separating them. This token is known to BERT, and used in its next sentence prediction task. Here, the token is used to separate different posts within a thread. After concatenation, the pretrained BERT model is used to obtain a vector representation of each token. Then, it is passed through a BiLSTM layer before being fed to the NeXtVLAD component. At this point, NeXtVLAD, as a parametric intelligent pooling and aggregation layer, represents the whole Tweet thread as a $K \times N$ matrix, which is finally flattened and fed to two dense layers with a softmax function to assign the predicted label. This architecture, based on the explanation provided by Lee et al. (2020), is presented in Figure 1.

Consider M input tokens, each represented by a vector of size N produced by the language model and further tuned by the BiLSTM layer (*e.g.*, $N = 1024$ for BERT$_{\text{Large}}$). We denote these tokens by x_t, $t \in \{1, ..., M\}$. Each x_t is expanded to $\dot{x}_t$ with shape $(1, \lambda N)$ and reshaped to $\tilde{x}_t$ with shape $(G, 1, \frac{\lambda N}{G})$. Then, the (1) soft assignment of $\tilde{x}_t^g$ to the cluster k, and (2) the attention over groups, are computed as

$$\alpha_{gk}(\dot{x}_t) = \frac{e^{w_{gk}^T \dot{x}_t + b_{gk}}}{\sum_{s=1}^{K} e^{w_{gk}^T \dot{x}_t + b_{gk}}} \qquad (1)$$

$$\text{and} \quad \alpha_g(\dot{x}_t) = \sigma(w_g^T \dot{x}_t + b_g). \qquad (2)$$

The locally aggregated feature vectors (*i.e.*, the VLAD vectors) are generated by computing the product of the attention, assignment, and the difference from the cluster center

$$v_{tki}^g = \alpha_g(\dot{x}_t)\alpha_{gk}(\dot{x}_t)(\tilde{x}_{ti}^g - c_{ki}).$$

Finally, the entire thread is represented by

$$r_{ki} = \sum_{t,g} v_{tki}^g.$$

In the above equations, t, g, k, and i iterate over tokens, groups, clusters, and vector elements respectively, while w and b denote the weight and bias parameters of the linear transformations in the fully-connected layers.

5 Experiments and Results

We delve into several modifications of the model, as well as various hyperparameter settings, in order to investigate how much effect the NeXtVLAD component has on the sarcasm detection task. Our experiments initially use the same training configuration as Lee et al. (2020), before exploring further.

Since Lee et al. (2020) employ additional unpublished data, an exact reproduction of the experiments is not possible. Moreover, the partition of the corpus into training and validation set is left unspecified. Thus, their results reported on the validation set are not truly comparable. Some hyperparameter settings, like the number of epochs for training, are also omitted from their report. However, the primary aim of this work is not to focus on reproduction of the results, but to determine what role the NeXtVLAD component played in the excellent final F_1 score of 93.1%.

	Validation set results				Test set results			
Model	**Precision**	**Recall**	**F-1**	**Accuracy**	**Precision**	**Recall**	**F-1**	**Accuracy**
BERT$_{\text{Large-Cased}}$	0.75	0.84	0.80	0.79	0.71	0.78	0.74	0.73
BERT$_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD	***0.74***	***0.84***	***0.79***	***0.78***	***0.71***	***0.77***	***0.74***	***0.72***
BERT$_{\text{Large-Cased}}$ + NeXtVLAD	0.71	0.82	0.76	0.74	0.69	0.77	0.73	0.71
BERT$_{\text{Large-Cased}}$ + BiLSTM	0.76	0.82	0.79	0.79	0.71	0.74	0.72	0.72
BERT$_{\text{Large-Cased}}$ + KimCNN + NeXtVLAD	0.74	0.84	0.79	0.78	0.72	0.82	0.77	0.75
BERT$_{\text{Large-Cased}}$ + OurCNN + NeXtVLAD	0.77	0.71	0.74	0.76	0.69	0.79	0.74	0.72
CTBERTv2	0.76	0.83	0.80	0.79	0.72	0.76	0.74	0.73
CTBERTv2 + BiLSTM + NeXtVLAD	0.72	0.85	0.78	0.77	0.71	0.79	0.75	0.73
BERT$_{\text{Large-Cased}}$ (DE)	0.81	0.85	0.83	0.82	0.72	0.73	0.73	0.72
BERT$_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD (DE)	0.79	0.84	0.82	0.81	0.73	0.74	0.74	0.73
BERT$_{\text{Large-Uncased}}$ (DE)	0.79	0.83	0.81	0.81	0.73	0.73	0.73	0.73
BERT$_{\text{Large-Uncased}}$ + BiLSTM + NeXtVLAD (DE)	0.79	0.87	0.82	0.82	0.73	0.79	0.76	0.75
CTBERTv2 (DE)	0.78	0.83	0.80	0.80	0.75	0.77	0.76	0.75
CTBERTv2 + BiLSTM + NeXtVLAD (DE)	0.81	0.83	0.82	0.82	0.77	0.77	0.77	0.77
BERT$_{\text{Large-Cased}}$ (DE, LA)	0.79	0.84	0.81	0.87	0.73	0.75	0.74	0.82
BERT$_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD (DE, LA)	0.67	0.60	0.63	0.77	0.63	0.52	0.57	0.74
Ensemble of 3 [CTBERTv2 + BiLSTM + NeXtVLAD] (DE)	0.62	0.61	0.61	0.62	0.60	0.54	0.57	0.59

Table 3: Sarcasm detection results. Precision, recall, and F_1 are shown for "sarcasm", while the accuracy is averaged over both classes. Experiments with dataset expansion (DE) and label augmentation (LA) are also included. The model identical to Lee et al. (2020) (minus data augmentation and modification) is shown in bold italics.

The performance of different configurations are shown in Table 3. Our results are shown for the original FigLang test set as well as the one-fifth validation set we separated from training[2]. All the models have been trained for 8 epochs with a batch size of 4. We train the models for different number of epochs ranging from 3 to 30. Lee et al. (2020) mention the use of early stopping for their number of training epochs, which aims to prevent overfitting by monitoring the model performance on a held-out set at the end of each epoch, and stopping the training when performance starts to degrade. Their work, however, leaves out two hyperparameter values required for replication: *patience*, which controls the number of consecutive times it is acceptable for a model to not improve, and *delta*, the minimum threshold for differential improvement.

Without these, we follow Fomin et al. (2020) and apply early stopping with patience and delta set to 2 and 0, respectively. With early stopping, the number of optimal epochs varied, but even while setting the random states manually to make the configuration as deterministic as possible, repeated experiments showed optimal training to always vary between 5 to 12 epochs (a subset of the more comprehensive experiments we conducted, checking from 3 to 30 epochs). In our experiments, the BERT$_{\text{large-cased}}$ + BiLSTM + NeXtVLAD model is identical to Lee et al. (2020) (without their data augmentation and modification). The hyperparameters for this model are provided in Table 4. Since this model achieves the best F_1 score on the validation set with 8 training epochs, we fix the number of training epochs to be 8 for the other models as well.

[2]Our code and the choice of validation set are available at github.com/sinamps/nextvlad-for-nlp

In order to replicate the ensemble model discussed by Lee et al. (2020), threads with more than one context are used to create extra samples by removing the furthest context, one at a time, until only one context remains. In the experiments using this data expansion (DE), the thread in Table 1, for instance, gives rise to one additional sample, with only context 2 and the response. Then, a separate model is trained for each context length, and majority voting assigns the final label. We also conduct a series of experiments where the response Tweet is removed from each thread, and the remaining thread is considered non-sarcastic. These are indicated in Table 3 by LA (label augmentation).

To explore further, we record the performance for all training epochs on the validation set. Table 5 shows the accuracy for epochs 2 to 8, for the model proposed by Lee et al. (2020) (the first configuration in Table 3). We compute the accuracy and F_1 score for up to 30 training epochs. A comparison of the best scores from the models that employ NeXtVLAD with the ones that do not, we find no statistically significant improvement.

We also include additional experiments that replace BiLSTM with convolution layers. We use KimCNN (Kim, 2014) as well as a custom CNN (simply called OurCNN in Table 3) with filters

Hyperparameter	**Value**
K	128
G	8
λ (expansion)	4
M	512
N	1024
Context Gating's dropout rate	0.5
BiLSTM's dropout rate	0.25
# of epochs	8
Batch size	4
Initial learning rate	10^{-6}

Table 4: The general hyperparameters for our implementation of $\text{BERT}_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD.

that always cover one response token with various number of context tokens. Appendix A provides a discussion of our custom CNN. These variations, too, however, do not outperform the baseline results obtained through BERT alone.

5.1 Discussion

In image/video processing, a large number of low-dimensional descriptors extracted from the original high-dimensional image (such as SIFT vectors of size 128) are fed to NeXtVLAD. In NLP applications, however, the token vectors have a much higher dimension. It is possible that this is why the subspace representation does not provide any advantage over the original vector representation. Another possibility is that unlike images or videos, sub-vector representations of tokens do not form meaningful units in natural language tasks, and thus, the low-dimensional split actually hurts the learner. Our experiments also show that the use of domain-specific models like CTBERT (Müller et al., 2020) offer comparable performance, but reach their best results in fewer epochs of training.

We feel that it is important to distinguish the components of a complex NLP pipeline that contribute to improvements in downstream tasks, from other components in the pipeline. While stopping short of providing explainability to a deep learning system, this type of investigation can, at the very least, provide *attribution* to specific components of NLP pipelines. In other words, it can help us identify *which* parts of a pipeline are primarily responsible for improvements in a downstream task.

Such attributions can help us build comparable systems that are significantly less resource-intensive. In our experiments, we were able to train models based on the $\text{BERT}_{\text{Large}}$ architecture with a 2-layer fully-connected classification head

	Accuracy for each epoch						
Model	2	3	4	5	6	7	8
w/o NeXtVLAD	0.73	0.77	0.78	0.78	0.78	0.78	0.79
w NeXtVLAD	0.51	0.49	0.49	0.76	0.77	0.77	0.78

Table 5: The validation set accuracy for training epochs 2 to 8 of the first model configuration from Table 3 (the first and second rows from Table 3).

with a batch size of 2 and sequence length of 512 on a single 12 GB GPU (NVidia GeForce GTX Titan X). But, with the addition of BiLSTM and NeXtVLAD, the same configuration was only able to fit a batch size of 1. For all the model configurations discussed in this paper, $\text{BERT}_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD required two 24 GB GPUs (Nvidia RTX 3090) to fit a batch size of 4.

6 Conclusion

We investigate the extent to which NeXtVLAD contributes to improved results in a recent sarcasm detection task, and find that it offers little in terms of additional benefits. Our conjecture at this point is, thus, that the 14% improvement achieved by Lee et al. (2020) must entirely be due to the natural language augmentation techniques used. Our work also indicates that local aggregators like NeXtVLAD are unlikely to offer significant benefits to tasks related to figurative language identification, but more empirical work is needed to confirm this hypothesis.

We hope that our insights can help future research in this direction by making it easier to channel their efforts into aspects of a pipeline that have tangible and attributable benefits to the final downstream NLP task.

Acknowledgements

This work was supported in part by funds from U.S. National Science Foundation (NSF) under award number CNS 2027750, CNS 1822118, and SES 1834597, and from NIST, Statnett, Cyber Risk Research, AMI, and ARL.

References

Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307. Institute of Electrical and Electronics Engineers.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. In *Proceedings of the Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 40–46. Association for Computational Linguistics.

Elisabeth Camp. 2012. Sarcasm, Pretense, and The Semantics/Pragmatics Distinction. *Noûs*, 46(4):587–634.

Matt Crane. 2018. Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

William Empson. 1947. *Seven Types of Ambiguity*, 2nd edition. Chatto and Windus, London.

Wael Etaiwi and Ghazi Naymat. 2017. The Impact of applying Different Preprocessing Steps on Review Spam Detection. *Procedia Computer Science*, 113:273–279.

V. Fomin, J. Anmol, S. Desroziers, J. Kriss, and A. Tejani. 2020. High-level library to help with training neural networks in PyTorch. `https://github.com/pytorch/ignite`.

Aniruddha Ghosh and Tony Veale. 2016. Fracking Sarcasm using Neural Network. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 161–169. Association for Computational Linguistics.

Debanjan Ghosh, Avijit Vajpayee, and Smaranda Muresan. 2020. A Report on the 2020 Sarcasm Detection Shared Task. In *Proceedings of the Workshop on Figurative Language Processing*, pages 1–11. Association for Computational Linguistics.

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating Local Descriptors into a Compact Image Representation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311. Institute of Electrical and Electronics Engineers.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

Polina Kuznetsova, Jianfu Chen, and Yejin Choi. 2013. Understanding and Quantifying Creativity in Lexical Composition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1258. Association for Computational Linguistics.

Hankyol Lee, Youngjae Yu, and Gunhee Kim. 2020. Augmenting Data for Sarcasm Detection with Unlabeled Conversation Context. In *Proceedings of the Workshop on Figurative Language Processing*, pages 12–17. Association for Computational Linguistics.

Rongcheng Lin, Jing Xiao, and Jianping Fan. 2018. NeXtVLAD: An Efficient Neural Network to Aggregate Frame-level Features for Large-scale Video Classification. In *Proceedings of the European Conference on Computer Vision Workshops*, pages 206–218. Springer.

Martin Müller, Marcel Salathé, and Per E Kummervold. 2020. COVID-Twitter-BERT: A Natural Language Processing Model to Analyse COVID-19 Content on Twitter. *arXiv preprint arXiv:2005.07503*.

Thomas Pfeiffer and Robert Hoffmann. 2009. Large-Scale Assessment of the Effect of Popularity on the Reliability of Research. *PLoS One*, 4(6):e5996.

Nils Reimers and Iryna Gurevych. 2017. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv preprint arXiv:1707.06799*.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in Twitter. In *Language Resources and Evaluation*, volume 47, pages 239–268. Springer Nature.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as Contrast between a Positive Sentiment and Negative Situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 704–714. Association for Computational Linguistics.

Josef Sivic and Andrew Zisserman. 2003. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477. Institute of Electrical and Electronics Engineers.

Tony Veale. 2011. Creative Language Retrieval: A Robust Hybrid of Information Retrieval and Linguistic Creativity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287. Association for Computational Linguistics.

A Appendix

To reduce the differences in the shape (*i.e.*, dimensions) and quantity of features fed to NeXtVLAD in Computer Vision and NLP, we designed a custom Convolutional Neural Network (CNN) to transform features into probably a more suitable space. In this section, we present the details of this custom

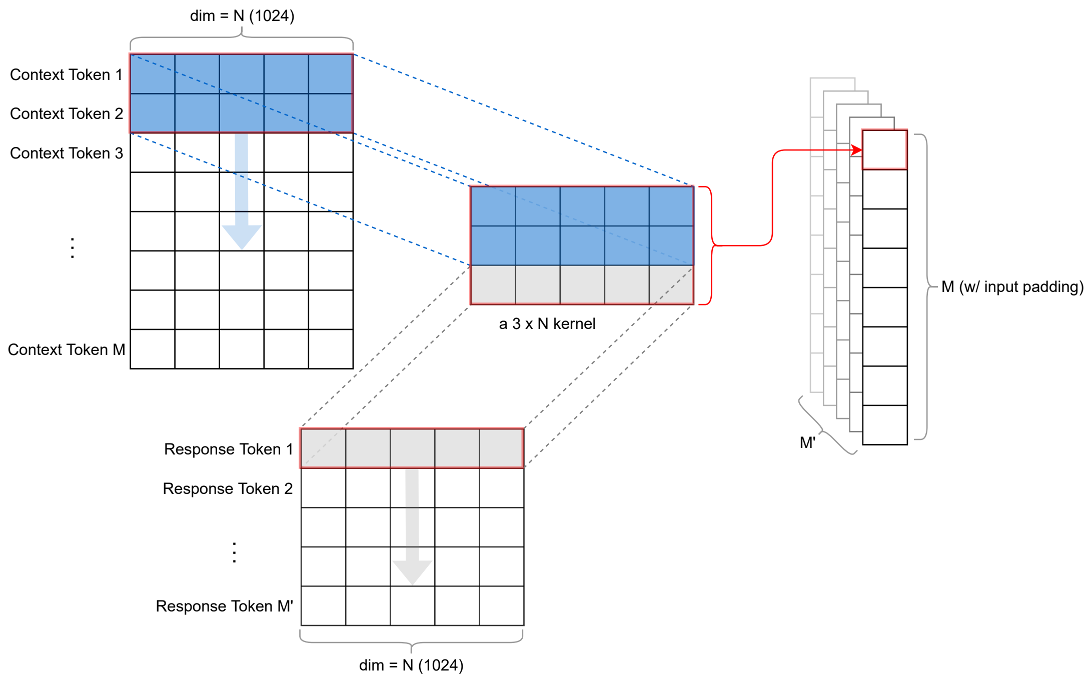

Figure 2: The custom CNN architecture for sarcasm detection in Twitter. *M* is the number of context tokens. *M'* is the number of response tokens. N is the token representation dimension.

CNN for extracting features for the NeXtVLAD layer. Figure 2 depicts the architecture of our CNN. First, we concatenate all the context Tweets and pass them to BERT to get the token representations and store them in a $M \times N$ matrix. The response Tweet also goes through the same process and is represented in a $M' \times N$ matrix. N is the dimension of the token representation vectors and M and M' denote the number of tokens in the contexts and response respectively. Each row in these matrices contains the vector representation of one token. Similar to KimCNN (Kim, 2014), we set the width of the kernel to the dimension of the token representation vector (N). But, distinct from KimCNN, our kernels are always applied to local areas from two distinct input matrices.

In our architecture, kernels only slide vertically to move over different tokens. To demonstrate, consider the kernel of size 3 in Figure 2. The first two rows of this kernel cover the first two tokens of the context matrix and the last row covers the first token in the response matrix. The inner product is computed and yields the first element in the first output vector. Then, the blue portion of the kernel slides downward and the computation repeats to yield the second element of the first output vector. When this sliding window reaches the end of the context matrix, the first output vector is complete. Now, the gray portion of the kernel slides downward on the response matrix and all previous steps repeat to generate the next output vector. This set of operations with F different kernels and by applying appropriate zero padding to the input, yields an output of shape (F, M', M) which is $(64, 100, 512)$ in our implementation. This output is rearranged and reshaped to shape $(M' \times M, F)$, which is much more similar to image/video features in shape and quantity. This is fed to NeXtVLAD in our sarcasm detection architecture. We use 64 kernels in our experiments with size 2, 3, 4, and 5 (16 kernels of each size; size only refers to the height of the kernel, since the width is fixed). In our implementation, the values are set as $F = 64$, $M = 512$, and $M' = 100$.