Identifying Aggression and Toxicity in Comments using Capsule Network

Saurabh Srivastava **Prerna Khurana** **Vartika Tewari**
TCS Research
{sriv.saurabh,prerna.khurana2,vartika.tewari}@tcs.com

Abstract

Aggression and related activities like trolling, hate speech etc. involve toxic comments in various forms. These are common scenarios in today's time and websites react by shutting down their comment sections. To tackle this, an algorithmic solution is preferred to human moderation which is slow and expensive. In this paper, we propose a single model capsule network with focal loss to achieve this task which is suitable for production environment. Our model achieves competitive results over other strong baseline methods, which show its effectiveness and that focal loss exhibits significant improvement in such cases where class imbalance is a regular issue. Additionally, we show that the problem of extensive data preprocessing, data augmentation can be tackled by capsule networks implicitly. We achieve an overall ROC AUC of 98.46 on Kaggle-toxic comment dataset and show that it beats other architectures by a good margin. As comments tend to be written in more than one language, and transliteration is a common problem, we further show that our model handles this effectively by applying our model on TRAC shared task dataset which contains comments in code-mixed Hindi-English.

1 Introduction

In today's time, with an ever increasing penetration of social media, news portals, blogs, QnA forums, and other websites that allow user interaction, users often end up inviting comments that are nasty, harrasing, insulting, toxic etc. This can have adverse effects on users, who then become victims of cyberbullying or online harrasment. An online survey carried out by the Pew Research Centre in 2017 states that 4 in 10 Americans have personally experienced online harrasment. Strikingly, 1 in 5 Americans have witnessed severe form of online harrasment like physical threats, stalking, sexual harrasment etc. There are several challenges associated with solving this kind of problem. First being the problem of class imbalance found in the dataset. Since such type of comments are sparse in nature, they introduce skewness in the dataset. There are several ways to handle this problem, however, we choose a more recent technique which modifies the standard cross entropy loss function known as Focal Loss (Lin et al., 2017). We will briefly describe how it helps in improving classifier performance. The next problem we want to address is that of data preprocessing. This is the most time consuming task and requires a good understanding of the data. However, we wish to minimise this process so as to have a good model with minimal preprocessing of the data.

Another frequently observed challenge is transliteration, which is often observed, especially, when we are working with text data from social networking websites. Users tend to speak in more than one language in the same statement. This leads to several *out of vocabulary* or OOV words for which the model would not have any word embedding. We use randomly initialised word embeddings in such a case and show how they can be trained during model training procedure such that it results in clusters of OOV words which have similar meaning in Hindi. We propose to tackle all the above described challenges using a single model as opposed to ensemble of several other models, which is a common practice in such competitive challenges. We also show that our proposed model can converge really quickly, hence the model can be trained in lesser time. This is essential when the model has to be deployed in a production environment where it requires retraining periodically.

Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying, pages 98–105
Santa Fe, USA, August 25, 2018.

2 Related Work

Early works in automated detection of abusive language made use of basic machine learning like Tf-Idf (Yin et al., 2009), SVM (Warner and Hirschberg, 2012), Naive Bayes, random forests, or logistic regression over a bag-of-ngrams and achieved limited success. Newer approaches include solving problems using deep learning architectures like CNNs (Kim, 2014; Zhang et al., 2015; Conneau et al., 2017b; Park and Fung, 2017) which just focus on spatial patterns or LSTMs which treat text as sequences (Tai et al., 2015; Mousa and Schuller, 2017). Another popular approach completely ignores the order of words but focuses on their compositions as a collection, like probabilistic topic modeling (Blei et al., 2003; Mcauliffe and Blei, 2008) and Earth Movers Distance based modeling (Kusner et al., 2015; Ye et al., 2017).

Recently Capsule Network (Sabour et al., 2017) has been used in text classification (Zhao et al., 2018).It makes use of the dynamic routing process to alleviate the disturbance of some noise capsules which may contain background information such as stop words and words that are unrelated to specific categories and show that capsule networks achieves significant improvement over strong baseline methods. As we focus to solve the problem of toxic comments and cyberbullying, we are confronted with the issue of large class imbalance. We use focal loss (Lin et al., 2017) to tackle it as it prevents the vast number of easy negatives from overwhelming the detector during training. Also, in the online space people tend to talk using different languages in the same comment and often use transliteration. We show that our model is suitable for such data as well.

3 Capsule Net for Classification

Proposed Model: The model proposed in (Zhao et al., 2018) has been used for the experimentation with an inclusion of Focal Loss (Lin et al., 2017) as a loss function to address the class imbalance problem. In our experiments we have compared performances of CNNs and RNNs as feature extractors and found that sentence representation obtained from RNNs performs better than representations obtained after applying convolution operation, although CNNs tends to perform better on short texts. The model consists of four layers:

(i) ***Word Embedding Layer:*** We represent every comment x_i, as a sequence of one-hot encoding of its words, $x_i = (w_1, w_2, ...w_n)$ of length n_{max}, which is the maximum length of the comment, with zero padding. Such a sequence becomes the input to the embedding layer. To represent word tokens several ideas like sparse representation or dense representation (Collobert and Weston, 2008; Bengio et al., 2003) have been proposed.

(ii) ***Feature Extraction Layer:*** This layer has been used to extract either n-grams feature at different position of a sentence through different filters (CNNs) or long term temporal dependencies within the sentence (RNNs). We use RNNs as feature extractors in our final model.

(iii) ***Capsule Layer:*** The Capsule layer is primarily composed of two sub-layers *Primary Capsule Layer* and *Convolutional Capsule Layer*. The primary capsule layer is supposed to capture the instantiated parameters of the inputs, for example, in case of texts local order of words and their semantic representation. Suppose we have $\hat{e}$ number of feature extractors, then the input to the Primary capsule layer will be $Z \in \mathbb{R}^{n \times e}$ (where n is the number of timesteps in RNNs). The primary capsules transform a scalar-output feature detector to vector-valued capsules to capture the instantiated features. Let d be the dimension of each capsule, then for each capsule $p_i \in \mathbb{R}^d$, where p denotes instantiated parameters set of a capsule (Sabour et al., 2017), we have $p_i = \mathbf{g}\left(WZ_i + \mathbf{b}\right)$, where Z_i is captured by RNNs in the feature extractor layer. Here, $\mathbf{g}$ is the nonlinear squash function which shrinks the small vectors to around 0 and large vectors around 1.

(iv) ***The Convolutional Capsule:*** The Conv layers capsules output a local grid of vectors to capsules in earlier layers using different transformation matrices for each capsule and grids member (Sabour et al., 2017).Capsule networks are trained using a dynamic routing algorithm that overlooks words that are not important or unrelated in the text, like stopwords and name mentions.

Model_Name	**Kaggle-toxic comment classification (ROC-AUC)**	**TRAC - 1 (English-FB) (Weighted F1)**	**TRAC - 1 (English-TW) (Weighted F1)**
CNN-multifilter	95.16	55.43	53.41
CNN-LSTM	96.85	62.20	47.68
Bi-directional LSTM with maxpool	97.35	59.79	51.146
FeedForward Attention Networks	97.42	57.43	55.49
Hierarchical ConvNets	97.95	51.38	50.43
Bi-LSTM, Logistic Regression	98.17	57.17	52.1
Bi-LSTM, xgboosted	98.19	57.33	52.31
Bi-LSTM with skip connections	98.20	61.78	51.98
Pre-trained LSTMs	98.25	60.18	58.7
CapsuleNet without Focal Loss	98.21	62.032	58.600
CapsuleNet with Focal Loss	**98.46**	**63.43**	**59.41**

Table 1: Comparison of several deep learning approaches with Capsule Net on the three datasets

Focal Loss: To handle the class imbalance problem, we have used Focal Loss which is given by the following formula :

$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$, where $p_t = \{ \begin{matrix} p & if\ y=1 \\ 1-p & else \end{matrix}$

γ is the focusing parameter which smoothens the rate at which easy examples are down weighted and, α is the weight assigned to the rare class.

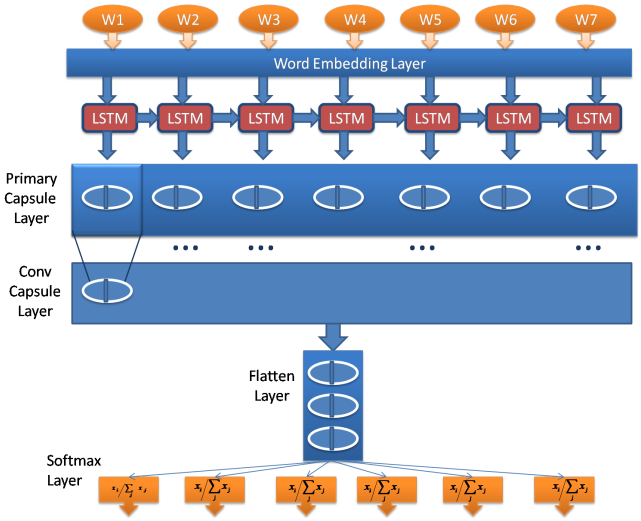


Figure 1: CapsNet with LSTMs as feature extractor

4 Experiments

In this section we attempt to describe different models that we have used for the classification process. We seek to answer the following questions: (1) Is combination of Capsules and focal loss the new apotheosis for toxic comment classification problems? (2) Can capsules solve the problem of OOV and transliteration implicitly ?

4.1 Datasets

4.1.1 Kaggle Toxic Comment Classification:

Recently, Kaggle hosted a competition named Toxic Comment Classification. This dataset has been contributed by Conversation AI, which is a research initiative founded by Jigsaw and Google. The task was comprised of calculating the log-likelihood of a sentence for the six classes, i.e., given a sentence calculate the probability of it belonging to six classes. The six different classes were toxic, severe toxic, obscene, threat, insult and identity hate.

4.1.2 TRAC dataset:

"First Shared Task on Aggression Identification" released a dataset for Aggression Identification. The task was to classify the comments into one of the three different classes Overtly Aggressive, Covertly Aggressive, and Non-aggressive. The train data was given in English and Hindi, where some of the comments in Hindi dataset were transliterated to English.

4.2 Data Preprocessing

For all our experiments, to show efficacy of our approach we kept the preprocessing as minimal as possible. Apart from word lowerization, tokenization, and punctuation removal we didn't perform any other activity.

4.3 Baseline Algorithms

We evaluate and compare our model with several strong baseline methods including: LSTM with Max-pool (Lai et al., 2015), Attention networks (Raffel and Ellis, 2015), Pre-trained LSTMs (Dai and Le, 2015), Hierarchical ConvNet (Conneau et al., 2017a), Bi-LSTM with Skip-connections, variation of CNN-LSTM (Wang et al., 2016), CNN-multifilter (Kim, 2014), Bi-LSTM with xgboost and logistic regression. We experiment with these models on three datasets. The models were first evaluated on Kaggle competition for Toxic Comment Classification. All the model parameters and attributes were decided on the basis of our best performing model, and were kept same for the rest of experimentations and datasets.

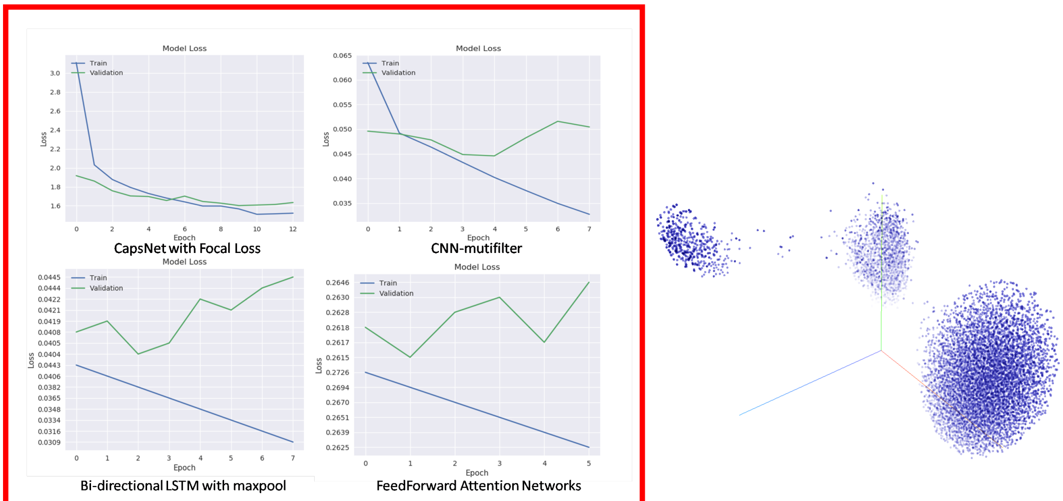
4.4 Model Training

For all our experiments we have used pre-trained embeddings for each word token obtained from (Joulin et al., 2016). We have also exploited (Pennington et al., 2014), (Mikolov et al., 2013), random and manually trained embeddings for initialization. After experimentation, fasttext embeddings with dimension of 300 were found to perform better than rest of the initialization process. In our experiments we observed that RMSProp (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2014) as an optimizer works well for training RNNs and CNNs respectively and used this throughout. The learning rate was kept between [.1 and .001]. For CNNs, number of Kernels was chosen from the range [128, 256, 512] and the LSTM units were selected from the range [128, 256]. In all of our experiments with the proposed model only a single layer for feature extraction was used. Number of capsules was varied from [8, 10, 16], the vector length of 8 for each capsule was found to be the best, and the dropout values for RNNs were taken as per suggestions from (Zaremba et al., 2014). The α and γ values in focal loss were experimented for [1.5, 2, 2.5, 3, 3.5] and [.2, .25, .3] and finally α = 2 and γ=0.25 were taken.

5 Results and Discussions

The proposed CapsNet architecture was able to beat other strong baseline algorithms with reasonable difference in accuracies with minimal preprocessing. We demonstrated that using focal loss along with CapsNet gave us .25 raise in the ROC-AUC for Kaggle's toxic comment classification and 1.39 and .80 gain in F1 scores on TRAC shared task dataset in English, from Facebook and Twitter comments respectively. All of our experiments were performed on NVIDIA Quadro M1200 4096 MB GPU system with 32 GB RAM and Intel i7 processor. The model took almost 33 minutes for an epoch to train which was faster in comparison with other models, with exception to the models using CNNs as feature extractors. For example, the second best performing model, which uses Pre-trained LSTM embeddings takes more than a day for the autoencoder to train and further 39+ minutes for each epoch. Hence, we can say that our model is viable for production environment.

We have tested the capability of the architecture to handle the OOV words or misspelled words. For this we used TRAC shared dataset, initialised the word embeddings randomly and trained the model for classification process. Next, we enabled the embeddings to be changed during training process which is mentioned as dynamic channel in (Kim, 2014) to let the model learn new embeddings. After training, we took the weights of embedding layer and plotted these embedings using Tensorboard (Abadi et al., 2015). From figure 2 we can see that the model is able to minimise the distance between the misspelled word



(a) Training and Validation Loss for Kaggle Toxic Comment Classification Dataset

(b) Clusters for word obtained after training

Figure 2

and is able to capture the relationship between transliterated words as in Table:2. We found that total of 3 clusters were formed after the experiment as shown in Fig:2b. We investigated these clusters and found that some of the highly used words in the comments belonged to certain classes. For example, one of the cluster contained more of neutral words, another cluster contained highly aggressive and abusive words, and the third cluster contained some toxic words along with place and country names related to one's origin which were used in some foul comments.
We show the capability of our model to tackle the problem of overfitting, as observed during training we see the model to have comparatively lower difference in training and validation loss than other models. Same can be seen from Fig:2a the loss margin difference doesn't change . We have shown that, not only our model has performed well on the classification task, it also has ability to generalise well and can learn good representation for word tokens.

NN to "politics"	NN to "bharat"
politic	bharatiya
politican	bhar
politico	mahabharata
politicize	bharti
politician	bhaskar

NN to "kut*e"(Hindi)
chu**ya
sa*le
tere
g**d
ma***rc**d

Table 2: Example of handling misspelt words and transliteration. NN : Nearest Neighbour

6 Conclusion and Future Work

In this work, we have proposed to automatically detect toxicity and aggression in comments, we show that with minimal preprocessing techniques we are able to achieve a good model performance and demonstrated how OOV words and semantic sense are learnt implicitly with random initialisation. We show the effectiveness of our proposed model against strong benchmark algorithms and that it outperforms others.
In this work, we did basic preprocessing of the data, however in future we intend to explore more preprocessing techniques for the dataset, like data augmentation using translation approaches and methods to deal with mispelled words. We further would examine the results of capsule net by visualising which words or phrases does the model correctly recognises for classification as opposed to benchmark algorithms. Also, we would like to examine the usage of focal loss with the rest of the baseline models.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017b. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007.

Jon D Mcauliffe and David M Blei. 2008. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.

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*.

Amr Mousa and Björn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1023–1032.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, EMNLP.

Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. 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 3856–3866. Curran Associates, Inc.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2428–2437.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Jianbo Ye, Yanran Li, Zhaohui Wu, James Z Wang, Wenjie Li, and Jia Li. 2017. Determining gains acquired from word embedding quantitatively using discrete distribution clustering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1847–1856.

Dawei Yin, Brian D. Davison, Zhenzhen Xue, Liangjie Hong, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on web 2.0.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*.

A Baseline Algorithm Descriptions

A.1 CNN Multifiter

The idea of applying CNNs for text classification was proposed in (Kim, 2014), where authors applied filters of different length to extract N-gram features from text. The authors tried static and dynamic embedding channels and concluded that the model with combination of both outperformed others. For our setting we found that filters of length [2, 3, 4] have outperformed other filter sizes, we tried various combinations from range [2, 5]. For activations we used Leaky ReLU, and performed Batch Normalization to stablize the data.

A.2 CNN LSTM

A joint architecture of CNNs and RNNs were proposed in (Wang et al., 2016), where the authors tried combination of CNNs with different RNNs like GRUs and LSTMs. In our experiment, we again used Leaky ReLU for CNNs activations, filter size of 3 was fixed for the experiments to decide the dropout values and other hyperparameters tuning.

A.3 Bi-directional LSTM with maxpool

In (Lai et al., 2015), authors took Max Over Time on the RNN representation of the input. Their model RNN outperformed other models in 3 out of 4 datasets. In our experiments, we fixed LSTM units to be 51, and rest of the parameters were decided on the basis of validation-data experiments.

A.4 Hierarchical ConvNets

Convolutional Neural Networks are known to perform well on short texts (Yin et al., 2017), in (Conneau et al., 2017a) authors proposed to concatenate representation at different levels of input sentence. The model was claimed to capture hierarchical abstractions of the input sentence. For our experiments, we fixed 128 kernels of size 2, 3, 4, 4 at 4 different levels. These values were decided after the experiments with different number of kernels and their sizes.

A.5 Bi-LSTMS with skip connections

In one of our experiments, the summary vector obtained from LSTMs was concatenated with the vector obtained after appying Max Over Time on the hidden state representation of the input. The intuition behind this was that, by passing most relevant features along with summary of the input to the softmax layer may enhance the clasification process. From the experiments we obtained competetive results using this model.

A.6 Pre-trained LSTMs

In (Dai and Le, 2015), authors claimed that by pretraining LSTMs on some related task as Auto-Encoder or as a Language Model, could optimize the stability of the LSTMs training process. The authors reported improvenemt in error rates by good margin in many tasks like, text classification on 20 Newsgroup, IMDB etc. For our experiments we gathered many related datasets like all of Wikimedia datasets (Wulczyn et al., 2017), TRAC shared dataset, IMDB movie reviews dataset. An autoencoder was trained on these datasets and the LSTMs from the encoder part were extracted and used in the classifcation task.