On Length Divergence Bias in Textual Matching Models

Lan Jiang[1], **Tianshu Lyu**[2], **Yankai Lin**[3], **Meng Chong**[2], **Xiaoyong Lyu**[2], **Dawei Yin**[2]
[1]Department of Automation, Tsinghua University
[2]Baidu Inc., Beijing, China
[3]Pattern Recognition Center, WeChat AI, Tencent Inc., China
jiangl20@mails.thu.edu yankailin@tencent.com
{lyutianshu, mengchong01, lvxiaoyong}@baidu.com yindawei@acm.org

Abstract

Despite the remarkable success deep models have achieved in Textual Matching (TM) tasks, it still remains unclear whether they truly understand language or measure the semantic similarity of texts by exploiting statistical bias in datasets. In this work, we provide a new perspective to study this issue — via the length divergence bias. We find the length divergence heuristic widely exists in prevalent TM datasets, providing direct cues for prediction. To determine whether TM models have adopted such heuristic, we introduce an adversarial evaluation scheme which invalidates the heuristic. In this adversarial setting, all TM models perform worse, indicating they have indeed adopted this heuristic. Through a well-designed probing experiment, we empirically validate that the bias of TM models can be attributed in part to extracting the text length information during training. To alleviate the length divergence bias, we propose an adversarial training method. The results demonstrate we successfully improve the robustness and generalization ability of models at the same time.

1 Introduction

Textual matching is a crucial component in various NLP applications, such as information retrieval (Li and Xu, 2014), question answering (Shen and Lapata, 2007) and duplicate detection (Bilenko and Mooney, 2003). Given a pair of texts, the goal is to determine the semantic similarity between them. A lot of deep models (Chen et al., 2017; Wang et al., 2017; Pang et al., 2016; Guo et al., 2019; Wan et al., 2016) have achieved excellent performance on various TM benchmarks.

However, recent work has found that current models are prone to adopting shallow heuristics in the datasets, rather than learning the underlying linguistics that they are intended to capture. This issue has been documented across tasks in natural language understanding. In natural language arguments, for example, Niven and Kao (2019) showed that model performance is inflated by spurious statistical cues. Similar heuristics arise in natural language inference (McCoy et al., 2019; Naik et al., 2018) and reading comprehension (Jia and Liang, 2017).

T_1 - Microsoft acquires Maluuba, a startup focused on general artificial intelligence. (**10**) T_2 - Microsoft has acquired Canadian startupMaluuba, a company founded by University of Waterloo grads Kaheer Suleman and Sam Pasupalak that also participated in. (**22**) Label: *Paraphrase* Output: *Non-paraphrase*
T_1 - Bill would cut off aid to countries that don't take back their illegal immigrant criminals. (**15**) T_2 - Common Sense law faces massive opposition supposing that Aid would be cut off to countries who refuse their citizens. (**19**) Label: *Non-paraphrase* Output: *Paraphrase*

Table 1: Examples for length divergence bias, originally from Twitter-URL. "Output" is the output label by ESIM trained on the original training set. Numbers in bold are the number of words each text consists of. Model is misled by the length divergence of two texts.

In this paper, we address this issue in the domain of textual matching. The focus of our work is on the length divergence bias — models tend to classify examples with high length divergence as negative and vice versa. Table 1 shows a single set of instances from Twitter-URL that demonstrates the length divergence bias.

We analyze current TM datasets and find that all of them follow specific length divergence distribution by labels. To determine whether TM models have employed this spurious pattern to facilitate their performance, we construct adversarial test sets which invalidate this heuristic and re-evaluate TM models. There is a performance drop on 14 out of total 16 combinations of models and tasks,

Findings of the Association for Computational Linguistics: ACL 2022, pages 4187 - 4193

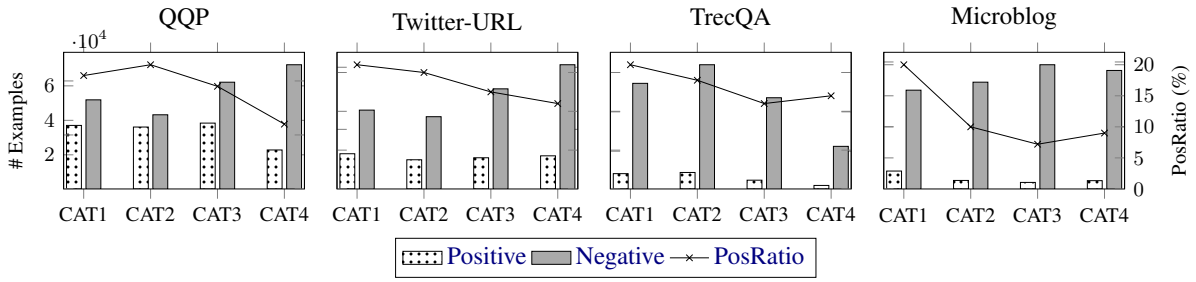

Figure 1: Length divergence distribution by labels across datasets. Bars represent the number of examples, corresponding to the left axis; polylines represent the ratio of positive examples, corresponding to the right axis.

suggesting their reliance on this heuristic.

Despite demonstrating the existence of length divergence bias, the underlying reason has not been well explained. By conducting the *SentLen* probing experiment (Conneau et al., 2018), we bridge this gap through revealing the text length information TM models have learned during training.

We finally explore a simple yet effective adversarial training method to correct the length divergence bias. The results show our approach not only reduces the bias but also improves the generalization ability of TM models. To encourage the development of TM models that understand semantics more precisely, we will release our code.

2 Datasets and Models

We select four well-known NLP and IR datasets as follows: **Quora Question Pairs (QQP)** (Wang et al., 2018), **Twitter-URL** (Lan et al., 2017), **TrecQA** (Wang et al., 2007), and **TREC Microblog 2013 (Microblog)** (Lin and Efron, 2013).

We study four models for textual matching tasks: **MatchPyramid** (Pang et al., 2016), **BiMPM** (Wang et al., 2017), **ESIM** (Chen et al., 2017) and **BERT** (Devlin et al., 2019). The four models above are representative in terms of neural architectures.

The detailed explanation for each dataset and model can be found in Appendix A.1 and A.2.

3 Length Divergence Heuristic in Current Datasets

In this section, we characterize existing datasets from the perspective of the length divergence between text pairs. We first formulate pairwise length divergence for NLP tasks and listwise length divergence for IR tasks, respectively.

Pairwise. Given two texts T_1 and T_2, their relative length divergence is defined as:

$$\Delta_{\text{rel}}\mathcal{L}(T_1, T_2) \coloneqq \frac{|\mathcal{L}_{T_1} - \mathcal{L}_{T_2}|}{\min(\mathcal{L}_{T_1}, \mathcal{L}_{T_2})}, \quad (1)$$

where

$$\mathcal{L}_T \coloneqq \#(\text{words in } T). \quad (2)$$

Listwise. In IR tasks, each example consists of a query Q and a list of documents $\boldsymbol{D}$ associated with it. We define the listwise relative length divergence with respect to Q as:

$$\Delta_{\text{rel}}\mathcal{L}(Q, \boldsymbol{D}) \coloneqq \frac{|\overline{\Delta_{\text{rel}}\mathcal{L}}(Q,\boldsymbol{D}^+) - \overline{\Delta_{\text{rel}}\mathcal{L}}(Q,\boldsymbol{D}^-)|}{\min(\overline{\Delta_{\text{rel}}\mathcal{L}}(Q,\boldsymbol{D}^+), \overline{\Delta_{\text{rel}}\mathcal{L}}(Q,\boldsymbol{D}^-))}, \quad (3)$$

$$\overline{\Delta_{\text{rel}}\mathcal{L}}(Q, \boldsymbol{D}^{+/-}) = \frac{\sum_{d^{+/-} \in \boldsymbol{D}^{+/-}} \Delta_{\text{rel}}\mathcal{L}(Q, d^{+/-})}{|\boldsymbol{D}^{+/-}|}, \quad (4)$$

where $\boldsymbol{D}^+$ is the set of relevant documents while $\boldsymbol{D}^-$ is irrelevant. For instances whose $\overline{\Delta_{\text{rel}}\mathcal{L}}(Q, \boldsymbol{D}^{+/-})$ does not exist or is equal to zero, we set $\Delta_{\text{rel}}\mathcal{L}(Q, \boldsymbol{D})$ to be a large number.

Based on the length divergence definition, we sort and split the training sets into quarters, namely `CAT1-4`, and examine length divergence distribution by labels for each dataset. Statistics are shown in Figure 1. We can see that all datasets suffer from the same problem: as the length divergence increases, the ratio of positive examples decreases. Overall, negative examples tend to have higher length divergence than positive ones, providing direct cues for label assignment.

4 Length Divergence Bias in TM Models

In this section, we demonstrate that existing TM models indeed employ the length divergence heuristic in datasets. Existing test sets are drawn from the same distribution as the training sets, which are overly lenient on models that rely on superficial cues. To provide a more robust assessment of TM models, we construct adversarial test sets by eliminating such heuristic.

4.1 Adversarial Test Sets

For two NLP datasets, adversarial test sets are built by the following steps: First, examples are sorted and split into four categories according to their

	Original					Adversarial				
	CAT1	CAT2	CAT3	CAT4	ALL	CAT1	CAT2	CAT3	CAT4	ALL
# Positive	4,055	3,967	4,280	2,583	14,885	3,385	2,882	4,013	2,583	12,863
# Negative	5,781	4,923	6,853	7,988	25,545	5,781	4,923	6,853	4,410	21,967
# Total	9,836	8,890	11,133	10,571	40,430	9,166	7,805	10,866	6,993	34,830
PosRatio	0.41	0.45	0.38	0.24	0.37	0.37	0.37	0.37	0.37	0.37

Table 2: Statistics of the original and adversarial test set on QQP task. Each category in the original test set is down-sampled to align with the average PosRatio.

length divergence. Second, we down-sample each category to align with the average PosRatio of the whole test set, i.e., the adversarial datasets we construct are subsets of theoriginal ones. Table 2 gives the details of the adversarial test set we build on QQP task, with a comparison of the original one.

The construction of IR datasets follows the same first step as NLP datasets. Considering random down-sampling may break the completion of query and its associated documents, in the second step, we discard the fourth category directly instead of down-sampling across all categories.

4.2 Re-evaluating TM Models

To examine whether TM models exploit the length divergence heuristic of existing datasets, models trained on the original training sets are evaluated on the original and adversarial test sets, respectively. We provide further details to facilitate reproducibility in Appendix A.3.

Results. The results are shown in Table 3. Overall, almost all models have a performance drop on all datasets (14 out of total 16 combinations). It seems that MatchPyramid captures the richest length divergence cues, as its performance drops most dramatically. BiMPM and ESIM both perform worse on the adversarial test sets except for one task. Although BERT outperforms other models, it cannot maintain its performance under adversarial evaluation either.

Moreover, we explore how the recall varies with the length divergence of examples. We report the recall for four length divergence categories of all models on QQP adversarial test set. Figure 2 shows that the recall declines across four categories, which indicates that TM models are more inclined to determine examples with high length divergence as negative, and vice versa.

We address that the adversarial evaluation scheme, which invalidates the length divergence heuristic, provides a more robust assessment for TM models. The above results are well apt with our intuitions about the length divergence bias: models

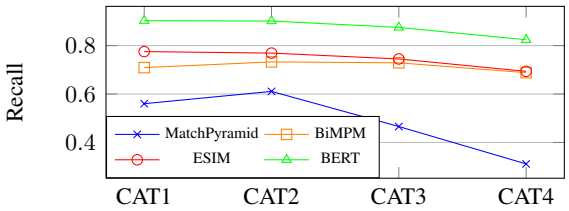

Figure 2: Recall of all models on the QQP adversarial test set. `CAT1` is the category with the lowest length divergence. Recall decreases across four categories, indicating TM models tend to determine examples with high length divergence as negative and vice versa.

do exploit some superficial cues about length divergence, instead of truly understanding the meaning of texts despite their good performance.

4.3 Probing Experiment

The adversarial evaluation has revealed the length divergence bias of TM models, but reason for this phenomenon is still unclear. In this section, we dig deeper into this problem.

Despite the variations of the architectures of TM models, all of them need to extract representation of texts first. The linguistic properties TM models capture have a direct effect on the downstream tasks. To explore what kind of information TM models have learned, we introduce a probing experiment using representations produced by TM models to predict the length of texts. We conduct the *SentLen* task in SentEval (Conneau et al., 2018), which is a 6-way classification task performed by a simple MLP with Sigmoid activation. As MatchPyramid cannot produce representation of a single text, we do not include it in this probing experiment. BiMPM and ESIM model both employ a BiLSTM encoder. We select the maximum value over each dimension of the hidden units as text representations, and use untrained encoders with random weights as the baseline. As BERT uses the output of the first token (`[CLS]` token) to make classification, we report the classification result using `[CLS]` token representations. We use the pre-trained model without fine-tuning as baseline.

Results. From Table 4 we can see that BiLSTM

Datasets	Metrics	MatchPyramid		BiMPM		ESIM		BERT	
		Original	Adversarial	Original	Adversarial	Original	Adversarial	Original	Adversarial
QQP	Acc	70.18 (+6.29)	68.66 (+7.31)	81.52 (-0.08)	80.91 (+0.15)	82.15 (-0.50)	81.38 (-0.15)	83.51 (+1.34)	82.57 (+1.67)
	BA	66.00 (+8.41)	64.60 (+9.44)	79.43 (+0.63)	78.97 (+0.81)	80.62 (-1.23)	80.01 (-0.96)	84.46 (+0.77)	83.65 (+1.04)
Twitter-URL	macro-F1	72.28 (+0.36)	71.72 (+0.38)	77.94 (+0.20)	77.63 (+0.17)	76.42 (+0.90)	75.91 (+1.21)	80.30 (+0.49)	80.10 (+0.55)
	micro-F1	84.23 (-0.26)	83.99 (-0.25)	85.50 (+0.12)	85.33 (+0.08)	86.58 (-0.46)	86.33 (-0.32)	85.26 (+0.50)	85.12 (+0.54)
TrecQA	MAP	60.22 (+6.18)	57.88 (+8.20)	88.75 (+2.24)	91.64 (+2.10)	76.84 (+7.18)	77.74 (+8.26)	87.22 (+1.31)	89.61 (+0.35)
	MRR	48.42 (+3.31)	47.27 (+5.95)	67.27 (+0.07)	67.56 (+0.13)	63.74 (-0.46)	60.99 (+3.06)	67.76 (-0.59)	65.75 (+0.12)
Microblog	MAP	18.93 (+0.23)	15.75 (+0.65)	26.44 (+15.06)	25.30 (+12.13)	14.54 (+22.61)	17.84 (+12.01)	47.11 (+2.79)	38.15 (+1.76)

Table 3: Performances of four TM models on the original and adversarial test sets for four tasks. Models are first trained on the original training sets. Performances of all systems drop on the adversarial test sets compared to on the original test sets. Models are then re-trained on the adversarial training sets. Numbers in parentheses indicate absolute gains from adversarial training.

Models	QQP	Twitter-URL	TrecQA	Microblog
BiMPM				
Untrained	56.18	54.47	56.23	57.03
BiLSTM-max	63.85	64.66	57.35	57.64
ESIM				
Untrained	58.28	56.99	55.81	56.89
BiLSTM-max	65.83	65.43	66.59	66.06
BERT				
Untrained	72.63			
BERT-`[CLS]`	70.66	72.10	81.33	75.08

Table 4: Probing task accuracies. Classifier takes text representation produced by TM models as input.

encoder has a severe performance boost across all datasets after training, which indicates that BiMPM and ESIM extract rich text length information during training. Compared to untrained BiLSTM encoder, BERT achieves a substantially better performance without fine-tuning. Interestingly, while BERT model suffers bias too, they are less pronounced, perhaps a benefit of the exposure to large corpus where the spurious patterns may not have held. It seems that pre-training enables BERT to extract relatively deeper linguistic properties and forget about superficial information.

The *SentLen* probing experiment reveals that TM models learn rich text length information during training, indicating the intrinsic reason why TM models suffer from the length divergence bias.

5 Length Divergence Bias Correction

In this section, we propose to correct the length divergence bias. As the model modification method usually has a significant cost and is inefficient, we apply adversarial training with bias-free training data. Our method is much more practical, lower cost, and easier to be implemented and adopted. We first construct the adversarial training sets in the same way as the adversarial test sets. We next re-train each model on the adversarial training sets and report their performance on two test sets.

Results. As presented in Table 3, performances improve for almost all models across four datasets except for one combination (ESIM on QQP). It is a little inspiring that adversarial training brings tremendous benefits to some models on IR tasks (MatchPyramid and ESIM on TrecQA dataset, BiMPM and ESIM on Microblog dataset). One possible explanation for this phenomenon is that, in IR tasks, the length divergence and class-imbalance are more severe than NLP tasks. While alleviating the length divergence bias of TM models, our method also makes models achieve better performances on the original test sets. Overall, the adversarial training not only successfully corrects the length divergence bias in TM models but also improves their generalization ability.

6 Conclusion

The inspiring success of deep models is accounted for by employment spurious heuristics in datasets, instead of truly understanding the language. In this work, we investigate the length divergence heuristic that textual matching models are prone to learn. We characterize current TM datasets and find that examples with high length divergence tend to have negative labels and vice versa. To provide a more robust assessment, we construct adversarial test sets, on which models using this heuristic are guaranteed to fail. Experiments show that almost all TM models perform worse on adversarial test sets, indicating they indeed exploit the length divergence cues. We then provide a deeper insight by conducting the *SentLen* probing experiment. TM models are shown to learn rich text length information during training, which accounts for the length divergence bias. Finally, we propose a simple yet effective adversarial training method to alleviate the length divergence bias in TM models. It's a little inspiring that our approach improves models' robustness and generalization ability at the same time. Overall, our results indicate that, there is still substantial room towards TM models which

understand language more precisely.

References

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 39–48, New York, NY, USA. Association for Computing Machinery.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

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.

Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. Matchzoo: A learning, practicing, and developing system for neural text matching. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1297–1300, New York, NY, USA. ACM.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A Continuously Growing Dataset of Sentential Paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Hang Li and Jun Xu. 2014. Semantic matching in search. *Found. Trends Inf. Retr.*, 7(5):343–469.

Jimmy J. Lin and Miles Efron. 2013. Overview of the TREC-2013 microblog track. In *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*, volume 500-302 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. 2019. Probing Neural Network Comprehension of Natural Language Arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2793–2799. AAAI Press. Event-place: Phoenix, Arizona.

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.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1913–1916, New York, NY, USA. Association for Computing Machinery.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2019. Multi-perspective relevance matching with hierarchical convnets for social media search. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc. Event-place: Long Beach, California, USA.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2835–2841. AAAI Press.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4144–4150, Melbourne, Australia. International Joint Conferences on Artificial Intelligence Organization.

Zhiguo Wang and Abraham Ittycheriah. 2015. FAQ-based Question Answering via Word Alignment. *arXiv e-prints*, page arXiv:1507.02628.

A Appendix

A.1 Datasets Details

Here, we provide details for the datasets we use.

Quora Question Pairs (QQP) (Wang et al., 2018) is a widely used benchmark in semantic matching. Each pair is annotated with a binary label indicating whether the two texts are paraphrases or not. We use split QQP from GLUE (Wang et al., 2018) benchmark with 363,849 examples for training and 40,430 for testing. We report accuracy (Acc) and balanced accuracy (BA).

Twitter-URL (Lan et al., 2017) is a sentence-level paraphrases dataset collected from tweets with 42,200 examples for training and 9334 for testing. For each pair, there are 6 raw annotations given by human raters. We perform the data preprocessing followed the author's notes.[1] We report macro-F1 and micro-F1.

TrecQA (Wang et al., 2007) is a widely used benchmark for question answering. According to Rao et al. (2016), there are two versions of TrecQA: both have the same training set, but their test sets are different. We use the clean version (Wang and Ittycheriah, 2015) with 53,417 examples for training and 1117 for testing. We report mean average precision (MAP) and mean reciprocal rank (MRR).

TREC Microblog 2013 (Microblog)(Lin and Efron, 2013) is a task to rank candidate tweets by relevance to a short query. We use the version prepared by Rao et al. (2019) with 39,378 examples for training and 6814 for testing. We report MAP.

Despite the fact that these datasets differ in tasks (similarity scoring vs. paraphrase detection vs. answer selection vs. tweet search), we regard all of them as a binary classification task to predict the textual similarity between two texts.

A.2 Models Details

Here, we provide details for the models we use.

MatchPyramid (Pang et al., 2016) views the matching matrix between two texts as an image, and a CNN is employed to learn hierarchical matching patterns.

BiMPM (Wang et al., 2017) matches encoded text pairs in two directions with four matching strategies. To accelerate the training procedure, we discard the character-composed embedding of the original BiMPM.

ESIM (Chen et al., 2017) is a sequential inference model based on chain LSTMs. We use the base ESIM without ensembling with a TreeLSTM.

BERT (Devlin et al., 2019) is a transformer-based (Vaswani et al., 2017) pre-trained language model. Due to the limitation of computational resources, we use $\text{BERT}_{\text{TINY}}$, which is a compact BERT model with 2 layers and 128 hidden units.

MatchPyramid is based on CNN, BiMPM and ESIM on RNN, and $\text{BERT}_{\text{TINY}}$ on Transformers.

[1] https://github.com/lanwuwei/Twitter-URL-Corpus

A.3 Training Details

We provide further details about the evaluation in Section 4 to facilitate reproducibility. We implement baselines based on open-source reproduction.[2]

Parameters setting. For MatchPyramid, BiMPM and ESIM, we use 300-dimension GloVe word embeddings (Pennington et al., 2014), and keep the pre-trained embeddings fixed during training. Words not present in the set of pre-trained words are initialized randomly. The kernel size of MatchPyramid is set to be 5×5 and 3×3. The dimension of hidden states of ESIM and BiMPM is set to be 128 and 100, respectively. They are trained using Adam (Kingma and Ba, 2015) with initial learning rate of $1e^{-4}$ and batch size of 64. For BERT, we use the implementation provided by the authors[3] and apply their default fine-tuning configuration.

Number of parameters in each model. The number of parameters in MatchPyramid is 15,053,562, of which 52,962 are trainable. The number of parameters in BiMPM is 15,890,202, of which 889,602 are trainable. The number of parameters in ESIM is 16,695,410, of which 1,694,810 are trainable. The number of parameters in $\text{BERT}_{\text{TINY}}$ is 4,386,178, and all of them are trainable.

[2]The open-source codes are available at https://github.com/pengshuang/Text-Similarity and https://github.com/airkid/MatchPyramid_torch

[3]Model is available at https://github.com/google-research/bert