FASTMATCH: Accelerating the Inference of BERT-based Text Matching

Shuai Pang*, Jianqiang Ma*, Zeyu Yan, Yang Zhang, Jianping Shen
{pangshuai550,majianqiang554,yanzeyu751,zhangyang147,shenjianping324}@pingan.com.cn
AI Team, Ping An Life Insurance Company of China, Ltd

Abstract

Recently, pre-trained language models such as BERT have shown state-of-the-art accuracies in text matching. When being applied to IR (or QA), the BERT-based matching models need to *online* calculate the representations and interactions for all query-candidate pairs. The high inference cost has prohibited the deployments of BERT-based matching models in many practical applications. To address this issue, we propose a novel BERT-based text matching model, in which the representations and the interactions are decoupled. Then, the representations of the candidates can be calculated and stored offline, and directly retrieved during the online matching phase. To conduct the interactions and generate final matching scores, a lightweight attention network is designed. Experiments based on several large scale text matching datasets show that the proposed model, called FASTMATCH, can achieve up to 100X speed-up to BERT and RoBERTa at the online matching phase, while keeping more up to 98.7% of the performance.

1 Introduction

Many natural language processing tasks, such as answer selection (Nakov et al., 2016) in question answering, query-document matching (Li and Xu, 2014) in information retrieval, natural language inference (Parikh et al., 2016) and paraphrase identification (Yin et al., 2015) can be formulated as *text matching*. Typical text matching model takes a pair of sentences (texts) as input and outputs their similarity score or label. Recently large Pre-trained Language Models (PLM) like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) have achieved state-of-the-art results in this field. However, these PLM-based models usually have high computational cost in both training and inference phase due to their very deep architectures with up to billions of parameters. In particular, the high latency caused by the intensive computation during the online inference makes the deployment of PLMs prohibitive for many real-world applications.

In particular, the high latency during online inference of BERT becomes crucial when text matching is applied to IR and QA, where a given *query* sentence needs to be matched to N ($> 10^2$) retrieved *candidates* sentences (documents). The PLM-based matching model has to repeatedly calculate the representations and interactions for all the N [query, candidate] pairs in an online manner. Denoting the time cost for matching each pair as T, the total time cost for processing a single query is $N \times T$, as shown in Figure 1 (a). For such *candidate-rich* text matching, the inference speed per query deteriorates by the increase of the number of candidates, N. Thus, how to accelerate the inference becomes a key challenge in applying PLM-based text matching models to IR/QA products.

To address this challenge, we propose FASTMATCH, an inference acceleration method for BERT-based text matching. Its key idea is *process decomposition*, where matching is decomposed into intra-sentence encoding with BERT and inter-sentence interaction with a lightweight *core matching model*. Since the encoding for all the candidates can be conducted offline, the inference procedure can be accelerated dramatically. Specifically, FASTMATCH works in a two-stage manner, as shown in Figure 1 (b). In the

*Equal contributions.

Proceedings of the 28th International Conference on Computational Linguistics, pages 6459–6469
Barcelona, Spain (Online), December 8-13, 2020

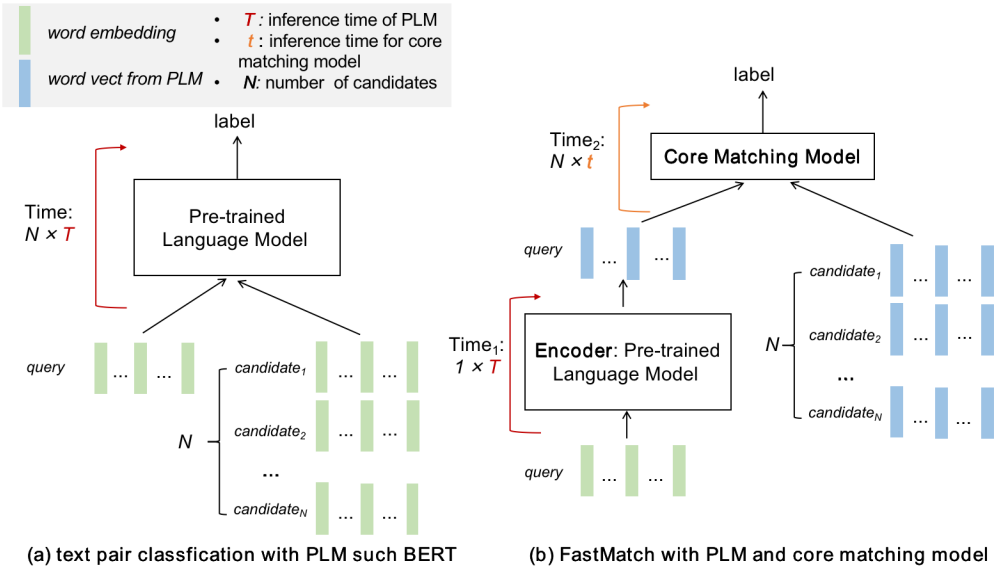

Figure 1: **Text matching with pre-trained language models (PLM)**

first stage, though the representation of the query is still computed online by BERT with the time cost of T, the representations of the N candidates can be pre-computed *offline* and cached with ignorable online retrieval time. In the second stage, the N pairs of [query, candidate] are fed to, instead of the original PLM, the core matching model, which takes time t to predict the label for each pair, arriving at a time cost of $N \times t$. To sum up, the total time cost for FASTMATCH is approximately $T + N \times t$. Thus, the speed-up ratio r is the ratio of the time cost of the PLM to that of FASTMATCH, as shown in (1). With large N, the speed-up ratio is dominated by T/t, i.e. the inference time of the PLM over that of the core matching model.

$$r = \frac{N \times T}{T + N \times t} = \frac{T}{\frac{T}{N} + t} \overset{N\uparrow}{\approx} \frac{T}{t} \tag{1}$$

While the query-independent and thus pre-computable encoding of candidates lays the foundation for efficient online inference, the real challenge lies in designing a substantially faster core matching model that can, at the same time, keep most of the PLM's accuracy. A naïve approach would be choosing an existing, lightweight text matching model to server the purpose. Unfortunately, using classic text matching architectures as the core matching model typically leads to dramatic accuracy drop compared with BERT-based model (Section 4.2). It turns out that it does not only require balancing speed and performance, but also calls for complementing BERT so that the whole architecture can get the most out of powerful contextualized representations.

In contrast with off-the-shelf models, the proposed core matching model (Section 3.1) is designed to take advantage of contextualized nature of BERT representations. Recall that in BERT, each token is recursively encoded by its context with a self-attention layer, using the representations of the most relevant words throughout the sentence from the previous layer. As a result, the word representations from a PLM carry a rich set of linguistic features at various granuarities. With such representations, a single interaction layer can largely capture the interactions between the query and candidate at different levels, a proven recipe for accurate matching (Hu et al., 2014). Based on such intuitions, we design an attention-based core matching model. In our model, BERT-based representations of query and candidate are fed to a cross-attention layer to mimic multi-level, query-candidate interactions. Then a self-attention layer is introduced to aggregate token-wise matching results into sentence-level matching result. Finally, we combine the matching results using yet another attention layer, after injecting diversities to interaction signals with a pooling layer. In practice, FASTMATCH can achieve more than 100X speed-up, while achieving comparable accuracy as the original PLM. The main contributions of this paper are three-fold:

- We propose FASTMATCH, a fast and accurate text matching method. To the best of our knowledge, FASTMATCH is the *first* process decomposition-based inference acceleration method for pre-trained language models.

- We propose a novel core matching model, which is the key to the superior results of FASTMATCH on accuracy-speed balance, compared to many established text matching models (Section 3.1).

- We show empirically that BERT and RoBERTa can be plugged into FASTMATCH, resulting in up to 100X speed-up for inference for text matching, while keeping more than 98.7% of the accuracy (Section 4.1, 4.4).

2 Related Work

Text matching Many text-based applications, such as Information Retrieval (IR), Question Answering (QA), Natural Language Inference (NLI) and Paraphrase Identification (PI) can be formalized as text matching. Table 1 summarises different paradigms in text matching. DSSM (Huang et al., 2013), CDSSM (Shen et al., 2014) and ARC-I (Hu et al., 2014) take a *representation-based* approach, which focuses on building the representation of the query and the candidates (documents) in a *dynamic* way. In particular, the representation of each candidate is built without considering the query. Thus, candidate representations can be pre-computed offline, leading to great reduction of computation at the online inference time. By contrast, *interaction-based* methods (Socher et al., 2011; Hu et al., 2014; Rocktäschel et al., 2015; Wan et al., 2016; Yin and Schütze, 2015; Yin et al., 2015; Pang et al., 2016; Chen et al., 2017; Kim et al., 2018) focuses on modeling the multi-level interactions of the query and candidate, while the representation of query and candidates are *static*, i.e. taking word embeddings of each token as input. Many representation-based methods fall into the matching-aggregation framework (Wan et al., 2016; Tan et al., 2018),which inspires the design of our core matching model. Interaction-based methods typically outperform representation-based ones, as the latter does not model the interactions between the query and candidate until their individual representations are fully built, thus taking the risk of missing important details.

Recently, BERT establishes itself as the new state-of-the-art in text matching and neural ranking (Qiao et al., 2019), thanks to the pre-training, as well as the modeling of *all-to-all* interactions between tokens in both query and candidate. In this case, the representations of candidate are built dynamically by taking the query into account, where the drawback is that the offline pre-computation of representations becomes infeasible. The proposed method, FASTMATCH, differs from BERT in that (1) its query-candidate interaction is conducted by a much faster core matching model rather than BERT itself and (2) the candidate representation is built independently from the query, thus can enjoy the benefits of offline computation of candidate representations, similar to representation-based method. However, unlike representation-based method, FASTMATCH explicitly computes query-candidate interactions in a multi-granularity manner. At the same time, FASTMATCH also distinguishes itself from interaction-based methods, as the interactions in FASTMATCH is based on dynamic, contextualized representation via BERT, instead of static word embeddings.

interaction / **candidate represent.**	*no*	*yes*
static	/	interaction-based
dynamic, w/o query	representation-based	FASTMATCH
dynamic, w/ query	/	BERT

Table 1: **Text matching paradigms**. *Interactions*: whether or not to model multi-level interactions between query and candidate; *candidate represent.*: whether the candidate representations are word embeddings (static) or are built by the model (dynamic), with or without encoding the query.

Model compression for inference acceleration The proposed method is built upon BERT and can also be seen as a specific way of inference acceleration for PLM, thus is related to other inference acceleration techniques. Since computational cost comes from model complexity, a family of inference acceleration methods take a *model compression* (Bucila et al., 2006) perspective: building a compact model to replace the original one. Most model compression work falls into three categories: (1) *quantization* (Gong et al., 2014); (2) *weights pruning* (Han et al., 2015): keeping the overall model architecture while reducing connections in certain layers; and (3) *knowledge distillation* (Sanh et al., 2019; Jiao et al., 2019): learning a simpler model to mimic the behavior of the original large model. For knowledge distillation with PLMs, DistilBERT (Sanh et al., 2019) reports a 60% faster model with more than 97% of kept accuracy on the GLUE benchmark (Wang et al., 2018). More recently, TinyBERT (Jiao et al., 2019) has suggested a new two-stage learning method for BERT, arriving at 9.4X speed-up, while keeping 96% of the accuracy on GLUE.Compared with above generic inference acceleration methods offering reasonable acceleration, FASTMATCH is tailored for text matching and can achieve more substantial speed-up up to 100X, as shown in Table 2.

Method	Acc.	Speed Up	Usage	Scenario
model compress.	96%	<9.7X	replace original PLM (pre-trained language model)	single & pair of sentences
FASTMATCH	96%	$>$ **100X**	fine-tune original PLM & train task-specific model	pair of sentences, $N > 1$

Table 2: **FASTMATCH v.s. model compression for inference acceleration**. ACC = relative accuracy w.r.t. PLM.

3 The FASTMATCH Method

The overall structure of FASTMATCH is depicted in Figure 1 (b). It combines a generic PLM with a task-specific core matching model, leading to a two-stage matching procedure. In order to conduct efficient matching, we propose an attention-based (Bahdanau et al., 2014) core matching model, the architecture of which is depicted in Figure 2. In the rest of this section, we go through each layer. We only show the equations for query q, as the readers can infer that for candidate p.

3.1 The core matching model

Cross attention layer for interaction It computes token-to-token interaction of the sentence pair. The cross attention works from query to candidate and vice versa, as it is well known bi-directional interactions outperform uni-directional ones. Figure 2 (a) shows the cross attention from q to p. For each token q_j in sentence q, its attention over the whole token sequence in p is computed as follows. First, the dot product of q_j with each token representation p_i in p, is computed as in (2), before feeding them a softmax function to obtain the weight for each p_i, denoting $\alpha^C_{i,j}$, as in (3). Then the weighted sum of all the p_i in sentence p is obtained as the new representation for q_j, denoted as q^C_j (equation 4), where C stands for the cross attention layer. Each token q_j is crossly encoded by the sum of its interactions with all the tokens in *the other* sentence.

$$e^C_{ij} = p_i \cdot q^T_j; \tag{2}$$

$$\alpha^C_{i,j} = \frac{exp\left(e^C_{i,j}\right)}{\sum_{k=1}^{|p|} exp\left(e^C_{k,j}\right)} \tag{3}$$

$$q^C_j = \sum_{i=1}^{|p|} \alpha^C_{i,j}\, p^C_i \tag{4}$$

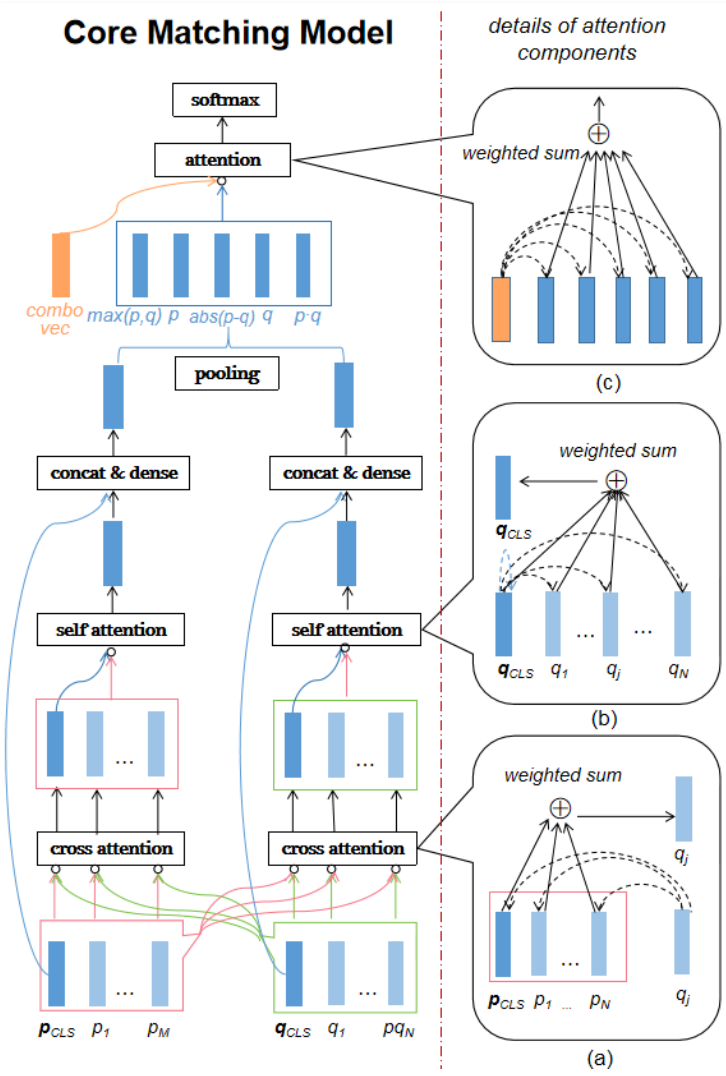

Figure 2: **The proposed core matching model.**

Self attention layer for aggregation The self attention layer aggregates the local interaction results at each token position to obtain a global vector for the whole sentence, capturing its interaction with the other sentence. As shown in Figure 2 (b), the computation procedure is otherwise quite standard, except the special treatment of q_{CLS}, which is the symbol representing the whole sentence, shown in *darker* blue in the figure. In the self-attention layer, we only compute the attention of q^C_{CLS} over all the tokens in the same sentence, including q^C_{CLS} itself, The intuition is that, the representation given by the cross attention layer for q^C_{CLS} summarize the interaction between the full sentence q and all the tokens in the other sentence. Therefore, the token-wise interactions that are consistent with q^C_{CLS} should be given more weights.

Concatenation and dense layers The output vector of the self-attention layer, q^S_{CLS} summarizes the interaction from q. It is then concatenated with q_{CLS}, the sentence representation given by the PLM encoder. The concatenated vector is fed to a dense layer with the *ReLu* non-linearity, resulting vector h_q.

Pooling layer for diversity Capturing the inter-sentence interaction with the cross layer is efficient but may lack diversity in terms of coverage for interaction patterns. Methods like (Wang et al., 2017; Tan et al., 2018) rely on multiple interaction layers in parallel to solve this problem. Since computational cost is critical, we stick to one cross-attention layer for interaction, while adding a pooling layer to foster the diversity after having the global interaction vectors via the self attention layer. As shown in the upper left of Figure 2, the pooling consists of five element-wise filters that are applied to each dimension of the two

input vectors, which are detailed in the following equation. We will see that these filters are commonly used metrics for capturing the sentence-pair relations in (Chen et al., 2017; Tay et al., 2017).

$$v_i = h_q,\ \ h_p,\ \ h_q \odot h_p,\ \ max(h_q, h_p),\ \ abs(h_q - h_p) \tag{5}$$

Attention layer for combining pooling result Pooling results v_i ($i = 1$ to 5) are summed up with the third attention layer shown in Figure 2 (c). The advantage of using attention is faster computation, compared with alternatives such as concatenating the pooling results and feeding to a dense layer. This final attention layer is similar to the self-attention layer, except that the attention is w.r.t. a *combo vector* randomly-initialized and optimized during training. The combo vector learns the weights for different pooling output to be summed up. Finally, the obtained vector v is fed to softmax layer for predicting the output label.

Training The core matching model is trained to optimize w.r.t. the final label accuracy of the sentence pairs, with cross entropy loss for classification or mean squared error loss for regression. Training of the core matching model is conducted *jointly* with fine-tuning the PLM encoder.

3.2 Time Complexity and speed-up effect

Speed-up effect As already discussed in Introduction, the vanilla sentence pair classification with the PLM needs to conduct N pair-wise classification each costing T, resulting a total time cost of $N \times T$, while the proposed FASTMATCH method has the total time cost of $T + N \times t$ for the N sentence pairs, leading to the speed-up ratio as shown in equation (1). Since $T > t$, by the increase of N, the $ratio$ steadily increases and approximates to T/t, the inference time of the PLM v.s. that of the core matching model.

Time complexity comparison For simplicity, we use d to denote the dimension of all the hidden layers as well as the dimension of word embeddings, while using l to denote maximum sentence length. In practice, we expect $l < d$ by an order of magnitude or so, taking values such as $l = 64$ and $d = 768$. The *cross-attention layer* has the time complexity of $O(l^2 \cdot d)$, as it conducts $l \times l$ cross-sentence token-wise dot product of dim-d. The *self-attention layer* needs a fraction of computation (by a factor of l) compared to the cross attention layer. The dense layer has the complexity of $O(d^2)$, as it involves one vector of dim $2d$ going through a $2d \times d$ matrix. The pooling and the last attention layer are relelatively simple and both has $O(d)$ complexity. In summary, The computation time is dominated by the cross attention layer costing $O(l^2 \cdot d)$ and the dense layers costing $O(d^2)$. Since $l^2 (\sim 10^4)$ is typically larger than d ($\sim 10^3$) in practice, the overall complexity of the core matching model can be approximated by $O_{core} = O(l^2 \cdot d)$. For PLMs such as BERT, the time complexity of each transformer block (Vaswani et al., 2017) is known to be $O_{BERT} = O(l \cdot d^2)$, making the whole PLM of the same time complexity with a rather *large co-efficient*. We can see that the time complexity of the core matching model, $O(l^2 d)$, is significantly lower than that of PLM, $O(ld^2)$. To see this, we take realistic values of $l = 64$ and $d = 768$, for which we have $ld^2 \sim 37.7 \times 10^6$ while $l^2 d \sim 3.1 \times 10^6$. The empirical number is in this range, as shown in Figure 3.

4 Experiments

This section presents FASTMATCH on a variety of text matching datasets. We first describe the setups and details before presenting main results in Section 4.1 with two standard QA datasets. In Section 4.2 the choice of core matching model is discussed. Ablation study comes in Section 4.3. And in Section 4.4 auxiliary evaluation of GLUE is conducted, where FASTMATCH achieves competitive results compared with model compression techniques for inference acceleration.

We use a batch size 32-64, with max sequence length of 64. Max training iteration is 12 with early stopping based on the dev set. First 10% steps are used for warming up. Adam (Kingma and Ba, 2014) optimizer is used with initial learning rate 1×10^{-5}. BERT-base and RoBERTa-base are used as encoders for FASTMATCH. Both BERT-base/RoBERTa-base have 12 layers, with 12 heads and hidden size $d = 768$. All models were trained in a single-task manner, with no ensemble or multi-task learning

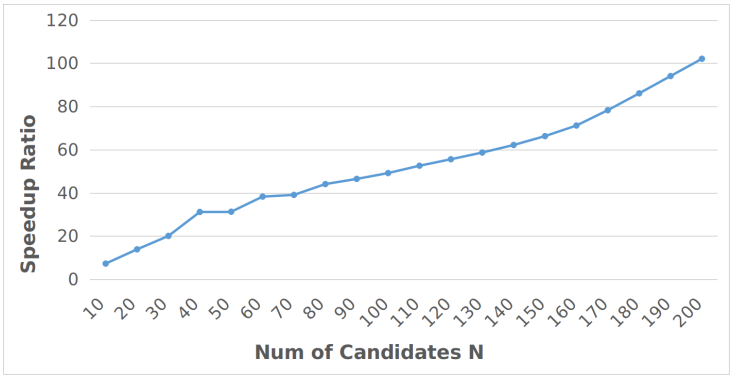

Figure 3: **Speed-up ratio v.s. candidate num. N**

employed. All experiments were conducted on Intel Xeon Gold 6130 CPU@2.10GHz, Tesla V100 GPU and CUDA 10.1.

4.1 Main results

We first study the speedup *and* performance retaining characteristics of FASTMATCH on two QA datasets. *First* is the TREC-QA dataset (Wang et al., 2007), which is a widely used benchmark for *answer selection*, where the candidate best matches the query is selected. It is based on the Text REtrieval Conference (TREC) QA track (8-13), which consists of 1162/65/68 unique queries in train/dev/test set, respectively. Each query has different number of candidate answers, with the median number N being 85. The *second* is from Task 3 of SemEval 2016, community question answering (Nakov et al., 2016). It is compiled for the *question-external comment similarity* task. Each query has 10 related questions, and each question has 10 comments attached. That is, every query has a fixed number of 100 candidates. The task is to rank these 100 comments by relevance score to the given query. It has 267/50/70 queries in train/dev/test set, respectively.

Metrics We use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as the performance metrics. MAP and MRR are widely used in TREC and SemEval tasks, as well as research like [24, 33]. On each dataset, we train two versions of FASTMATCH, based on BERT and RoBERTa, respectively. Then we compare the performance and speedup of FASTMATCH and corresponding PLM, which is used conventionally as a sentence pair-wise classifier. In addition, we also fine-tune the SOTA BERT-based knowledge distillation model TinyBERT for comparison. The main results are illustrated in Table 1, showing that in each group FASTMATCH is on par with corresponding PLM, retaining up to 97.8%/98.7% of the original PLM on MAP/MRR. Meanwhile It achieves astonishing 47.2X and 49.2X speedups on TREC-QA and SemEval-2016, respectively. By contrast, SOTA compression-based method, TinyBERT can only offer a data-set invariant speed-up ratio of 9.7X, which is substantially lower than FASTMATCH. Moreover, the performance retaining capacity of TinyBERT is also surpassed by our model on both datasets. Such results suggest that FASTMATCH is very competitive for accelerating the inference of BERT-based text matching.

Model	TREC-QA			SemEval-2016		
	MAP	MRR	*Speedup*	MAP	MRR	*Speedup*
BERT-base	0.829	0.871	1X	0.474	0.559	1X
tinyBERT	0.749	0.774	9.7X	0.446	0.508	9.7X
FASTMATCH$_{BERT}$	**0.807**	**0.849**	**47.2X**	**0.462**	**0.539**	**49.2X**
RoBERTa	0.832	0.872	1X	0.479	0.569	1X
FASTMATCH$_{RoBERTa}$	0.814	0.861	47.2	0.447	0.512	49.2X

Table 3: **Main results**. Accuracy in percentage. Best results marked in bold.

Model	TREC-QA			SemEval-2016		
	MAP	MRR	Speedup	MAP	MRR	Speedup
BERT pairwise	0.829	0.871	1X	0.474	0.559	1X
BERT + cosine	0.681	0.711	**119.1X**	0.382	0.444	**125.6X**
BERT + MLP	0.692	0.718	78.3X	0.395	0.461	80.5X
BERT + MatchPyramid	0.745	0.798	49.6X	0.402	0.469	52.1X
BERT + DecAttend	0.752	0.791	48.4X	0.416	0.489	50.8X
FastMatch$_{BERT}$	**0.807**	**0.849**	47.2X	**0.462**	**0.539**	49.2X

Table 4: The choice of the core matching model. Best results shown in bold.

As discussed in Section 3.2, the speed-up effect of FASTMATCH is amplified when number of candidates N is large. The empirical speed-up ratio for BERT and RoBERTa (of the same architecture as BERT) is depicted in Figure 3. We expect to further improve the acceleration ratio with even larger N and stabilizes it at a certain number, which is determined by T=t as analyzed in Section 3.2. Unfortunately, due to resource limitations we are unable to plot the results beyond N = 200, where the speed-up ratio is already > 100.

4.2 The choice of core matching models

Having established the effectiveness of FASTMATCH for speeding up BERT while retaining most of the accuracy, we further verify the usefulness of the proposed core matching model. To this end we compare several FASTMATCH variants, the matching model of which are replaced by classic text matching architectures also on TREC-QA and SemEval-2016 dataset. Specifically, the matching model to be replaced including cosine similarity, Multi-Layer Perceptron(MLP) with one hidden layer, classic models like *MatchPyramid* (Pang et al., 2016) and *DecAttend* (Parikh et al., 2016). BERT-base is used as the encoder for all these variants. BERT-base model is also used as sentence-pair classifier for comparison. Results are shown in Table 4.2.

What makes a good core matching model? It turns out that our proposed core matching model dwarfs not only the cosine and MLP baselines, but also classic architectures such as MatchPyramid and DecAttend. It suggests that well-established methods are not inherently good choices for core matching models. The core matching model needs to make good use of BERT representations and work together with BERT. This is exactly the process decomposition approach that FASTMATCH takes. FASTMATCH utilizes BERT representations, which capture linguistic features at various granularities, and leverage cross- and self-attention layers to capture multi-level interactions. The effectiveness of attention is also shown by the performance of DecAttend, which ranks right below our model. We conjecture that our model may enjoy most advantages that transformers (Vaswani et al., 2017) have, because of their similarity in architecture. Our cross-attention and self-attention layers are special cases of the self-attention as in transformers, where our layers attend on the CLS token and tokens in the other sentence, respectively, instead of attending on each token in the sentence pair.

4.3 Ablation Study

Here we study the influence of components in FASTMATCH with same setup. Results are shown in Table 5. The cross- and self-attention layer work together as a mini matching aggregation, which is vital to FastMatch. Removing them leads to a dramatic accuracy drop. The pooling and pooling-attention layers are designed to foster diversity in matching patterns, the removal of which also leads to a significant drop. This shows the necessity of complementing the first two attention layers with the attentive pooling layers.

4.4 Auxiliary Results on GLUE

In previous subsections, FASTMATCH presents high acceleration ratio while retaining most of accuracy. The acceleration effect relies on the candidate number N as shown in Figure 3, thus needs no

Model	TREC-QA		SemEval 2016	
	MAP	MRR	MAP	MRR
FastMatch_BERT	0.807	0.849	0.462	0.539
-cross-attention-self-attention	0.741	0.787	0.412	0.488
-self-attention	0.772	0.809	0.424	0.497
-pooling -attention after pooling	0.787	0.818	0.422	0.496

Table 5: **Ablation Study**.

validation with different datasets. It is still interesting to see whether the high accuracy retaining characteristics can generalize to other text matching datasets. For this purpose, we choose the widely-used GLUE benchmark(Wang et al., 2018). Recall that FASTMATCH deals with sentence-pair tasks, thus we exclude single-sentence tasks CoLA and SST-2. Following (Devlin et al., 2019), we also exclude the problematic WNLI set, leaving three NLI datasets(MNLI, QNLI, RTE), two PI datasets (MPRC, QQP) and one textual similarity dataset (STS-B) for evaluation. We compare the accuracy-retaining results of FASTMATCH with SOTA inference acceleration methods, TinyBERT and DistilBERT[1]. We report the performance with standard metrics as in (Devlin et al., 2019) for FASTMATCH. We also calculate the average *relative accuracy* on all six datasets. FastMatch$_{RoBERTa}$ achieves the best among the four models, even surpassing BERT on the three NLI datasets, giving credit to FastMactch's accuracy retaining from RoBERTa. On relative accuracy, FASTMATCH consistently keeps most of the accuracy of the BERT and RoBERTa across all datasets, leading to average relative accuracies higher than 96%. This makes FastMach on-par with TinyBERT and significantly better than DistilBERT. It suggests that FastMatch is competitive on inference acceleration of PLM for text matching.

Model	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Rel.
	F1	F1	SCol	ACC	ACC	ACC	Acc.
DistilBERT	82.4	68.5	76.1	78.9	85.2	54.1	91.1
TinyBERT	**86.4**	**71.3**	79.9	82.5	87.7	62.9	**96.6**
BERT†	87.6	70.1	85.8	84.6	90.5	66.0	/
FASTMATCH$_{BERT}$	80.6	69.0	81.6	83.1	88.7	63.9	96.4
RoBERTa†	88.7	70.9	85.5	87.6	94.0	83.2	/
FASTMATCH$_{RoBERTa}$	82.0	69.2	**82.4**	**85.9**	**91.5**	**79.9**	96.3

Table 6: **Results on GLUE subsets**. *Scol*:Spearman's correlation coefficient. *Rel. Acc.*:relative accuracy.Best results marked in **bold**.†: reproduced results.

5 Conclusion and Future Work

To deal with the prohibitive online computational cost of PLM for real-world applications, we propose FASTMATCH, an inference acceleration method of BERT for text matching tasks with a rich set of candidates, such as answer selection in QA and document ranking in IR. FASTMATCH decouples text matching into representation encoding and interaction computation, where the candidates encoding are calculated and stored offline with ignorable online retrieval time. For online interaction-based matching, we introduce an attention-based core matching model, which has significant lower time complexity than BERT. Experiments show that FASTMATCH can achieve up to 100X speed-up to BERT and RoBERTa in online matching, while keeping most of their accuracy. FASTMATCH offers an alternative to model compression for inference acceleration with PLM. With superior speed-up ratio, it can be useful for other similar tasks. Moreover, as FASTMATCH is orthogonal to model compression, combining them together can yield more promising results.

[1]For DistilBERT, we use the re-produced results from TinyBERT, as the original results are on the *dev* set with different metrics, making the results incomparable.

Acknowledgements

We thank Jun Xu of RUC, Wenpeng Yin, Muhua Zhu, Benyou Wang as well as all the anonymous reviewers for their invaluable comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.

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, July. 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, June. Association for Computational Linguistics.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *ArXiv*, abs/1412.6115.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

Xiaoqi Jiao, Y. Yin, Lifeng Shang, Xin Jiang, Xusong Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *ArXiv*, abs/1909.10351.

Seonhoon Kim, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak. 2018. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *AAAI*.

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

Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information Retrieval*, 7:343–469.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of SemEval-2016*, pages 525–545, San Diego, California.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *AAAI*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November. Association for Computational Linguistics.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of bert in ranking. *ArXiv*, abs/1904.07531.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809. Curran Associates, Inc.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4411–4417. International Joint Conferences on Artificial Intelligence Organization, 7.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

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 *AAAI*, volume 16, pages 2835–2841.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

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, IJCAI-17*, pages 4144–4150.

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, November. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado, May–June. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL*, pages 259–272.