FastHybrid: A Hybrid Model for Efficient Answer Selection

Lidan Wang
IBM Watson
T.J. Watson Research Ctr.
Yorktown Heights, NY
wangli@us.ibm.com

Ming Tan
IBM Watson
T.J. Watson Research Ctr.
Yorktown Heights, NY
mingtan@us.ibm.com

Jiawei Han
Computer Science Dept.
UIUC
Urbana, IL
hanj@cs.uiuc.edu

Abstract

Answer selection is a core component in any question-answering systems. It aims to select correct answer sentences for a given question from a pool of candidate sentences. In recent years, many deep learning methods have been proposed and shown excellent results for this task. However, these methods typically require extensive parameter (and hyper-parameter) tuning, which gives rise to efficiency issues for large-scale datasets, and potentially makes them less portable across new datasets and domains (as re-tuning is usually required). In this paper, we propose an extremely efficient hybrid model (*FastHybrid*) that tackles the problem from both an accuracy and scalability point of view. *FastHybrid* is a light-weight model that requires little tuning and adaptation across different domains. It combines a fast deep model (which will be introduced in the method section) with an initial information retrieval model to effectively and efficiently handle answer selection. We introduce a new efficient attention mechanism in the hybrid model and demonstrate its effectiveness on several QA datasets. Experimental results show that although the hybrid uses no training data, its accuracy is often on-par with supervised deep learning techniques, while significantly reducing training and tuning costs across different domains.

1 Introduction

Open-domain question answering (QA) aims to serve a user's information request by returning a list of direct answers. This problem has been receiving an increasingly amount of attention in the NLP and machine learning communities in the recent years (Ferrucci et al., 2012; Etzioni et al., 2011). Answer sentence selection (Yih et al., 2013; Tan et al., 2016; Yu et al., 2014; Severyn et al., 2013), which, given a user question, returns the correct sentences that contain the exact answer, is a core component in QA systems. The performance of QA systems critically depends on choosing the right candidate sentences which facilitate the extraction of final answers.

To be successful, in addition to accuracy, real-world systems must be scalable and select the most accurate answer sentences in a short amount of time. However, accuracy and speed/scalability are competing forces that often counteract each other. It is often the case that methods developed for improving accuracy incur moderate-to-large computational costs. For example, models based on neural networks have become very popular due to their strong accuracy for this task (Yu et al., 2014). However, they are typically slower at training and test time as compared to simple models, which may limit their use on very large datasets (Joulin et al., 2016). The speed issue is particularly important when working with new domains and datasets, as the model may have to be re-trained or adapted for the new dataset. Model re-training and adaptations, even with incremental techniques (Zhou et al., 2012; Chopra et al., 2013; Glorot et al., 2011) on state-of-the-art hardware (Chilimbi et al., 2014; Xing et al., 2015), could be prohibitively inefficient when working with time-critical applications or an impatient end-user – who may prefer a method with minimum (or no) training time spent, *and* getting similar accuracy as the expensively trained models.

On the other hand, classical information retrieval (IR) models are extremely fast as compared to the deep models, but may not be as quite accurate. IR models use fast word matching features (e.g., unigram, bigram overlaps between question and answer) to quickly score candidate sentences. Typically no training is necessary, since these simple models are fairly robust and can be applied to different datasets without modification (Tao et al., 2007; Bendersky et al., 2010; Buttcher et al., 2006). This property makes them attractive for scaling to large number of new domains and datasets.

In this paper, we take a step to eliminate the virtual dichotomy between accuracy and scalability/speed by developing a hybrid model that is simultaneously effective (as the deep models) and extremely efficient at training and test time (as the simple models). We name our proposed structure *FastHybrid*. We introduce the concept of a fast (deep) model in Section 3, then describe the *FastHybrid* model which combines IR with the proposed fast (deep) model for the best possible accuracy and training speed.

Unlike many other deep learning models, the fast deep model skips the intermediate convolution steps altogether, and directly operates on the raw word vectors coming from the answer and a slightly modified question text, constructed from a simple attention approach we propose in Section 3.2, to perform standard pooling operations. The simple attention in the fast deep model (Section 3.2) works by focusing explicitly on the question's influence on the answer with respect to the answer's representation. While seemingly, important information from convolution may get lost, however, with the new simple attention, our model often beats its expectations – not only does it perform well for time, but for accuracy equally.

The fast deep model is combined with an IR model via a hybrid structure to handle answer selection accurately and efficiently. The IR model is used to create an initial ranking of candidate sentences, and for the questions that cannot be handled well by IR, the fast deep model is applied. The hybrid structure leverages complementary strengths by IR and the fast model (both in terms of accuracy and speed), as will be discussed in Section 3.4.

Our model is nearly hyper-parameter/parameter-free, so it is extremely efficient (no training) for different datasets, and as a result, it can scale very easily to a large number of new domains and users. The remainder of the paper is organized as follows: we start with a discussion of related work, Section 3 describes our fast model and the hybrid approach, and our methods are evaluated in Section 4, before discussing future work and concluding.

2 Related work

In recent years, the problem of answer selection, which is a sub-problem in question-answering, has been getting a lot of attention in the research community (Yih et al., 2013; Tan et al., 2016; Yu et al., 2014; Severyn et al., 2013). Moving away from the shallow word-level features, these deep learning approaches focus on extracting important features from low-level representations (word embeddings) by using various types of deep neural networks (Graves et al., 2013; Hochreiter et al., 1997). The resulting models can effectively work at a semantic-level (i.e., can match a correct answer with a question even if they do not have any words in common, however semantically related). This can be viewed as a big improvement in comparison to the standard information retrieval approaches for the same task, which typically use hand-crafted word matching features.

However, from an efficiency's point of view, the deep models are very time-consuming to train. In particular, to achieve good success, the models typically require a large number of parameters and weights, making parameter (and hyper-parameter) tuning an expensive process for large-scale applications. While many remedies have been proposed to address the efficiency issue, ranging from developing fast training hardwares (Chilimbi et al., 2014) to parameter sharing and simplifying model structures, in comparison to standard simple IR approaches (Tao et al., 2007; Bendersky et al., 2010; Buttcher et al., 2006), the training efficiency and scalability of these methods still significantly lack behind.

This issue is particularly significant when we have to adapt the model repeatedly for new domains. While domain-adaptation techniques have been proposed for many deep learning models (Chopra et al., 2013; Zhou et al., 2012; Ganin et al., 2015; Sun et al., 2016), the focus and starting point of these work have largely been on how to adjust the models for accuracy, rather than from an accuracy and scalability point of view.

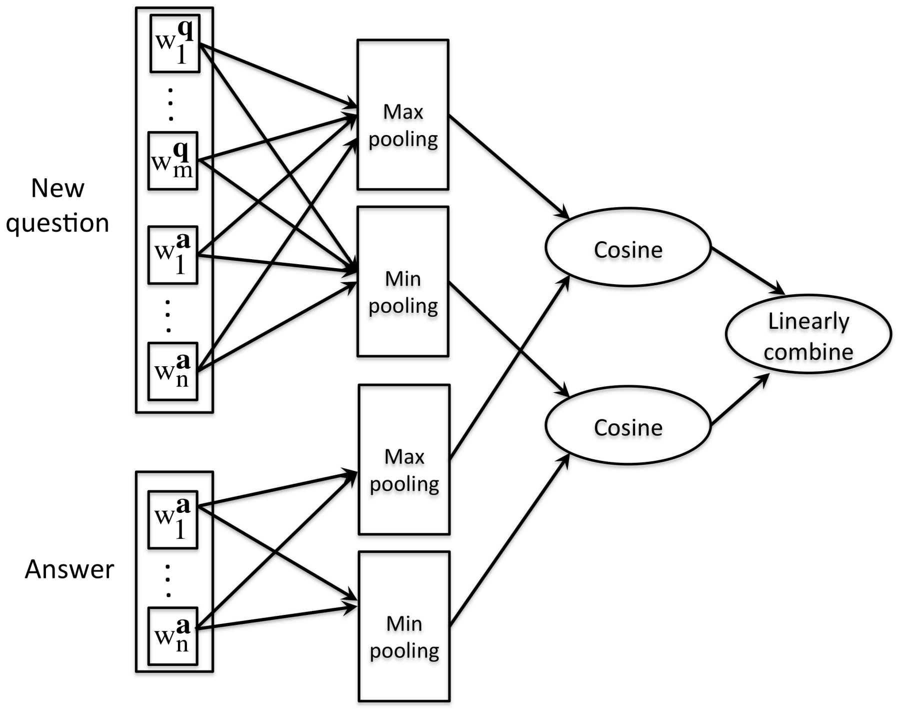


Figure 1: Model architecture for simplified deep learning model (a.k.a. fast model)

3 Methods

In this section, we describe our proposed hybrid model, *FastHybrid*, for effective and efficient answer selection. It combines a fast (deep) model[1] with an initial IR model to create an efficient and effective overall structure for this task.

A major area that distinguishes our work from the past deep learning models is that we directly operate on the raw word embedding vectors to perform standard aggregation operations (e.g., max, min-pooling) before computing similarity scores (Section 3.3). We completely discard the convolution operations used by many other deep models altogether. With the simple attention (Section 3.2) used by the fast model, our system is orders of magnitude faster in training than the standard deep models, while performing well for accuracy equally. We begin with a quick overview of the preliminary background on the Word2Vec embedding, then focus on the fast (deep) model in Section 3.2 and Section 3.3, before discussing the hybrid structure in Section 3.4.

3.1 Word2Vec Embedding

Recently (Mikolov et al., 2013) introduced *word2vec*, a word-embedding procedure. They use a shallow neural network language model to learn a vector representation for each word. More specifically, a neural network architecture (the skip-gram model) is proposed and consists of an input layer, a projection layer, and an output layer to predict words nearby. Each word vector is trained to maximize the log probability of neighboring words in a corpus. That is, given a sequence of words $w_1, \ldots, w_Z$,

$$\frac{1}{Z} = \sum_{z=1}^{Z} \sum_{i \in nb(z)} log\ p(w_i|w_z)$$

[1]We slightly abuse terminology and refer to it as a fast (deep) model. Note no convolution filters etc (representative of standard deep models) are used. We use this name mostly for comparison purpose with the state-of-the-arts deep learning models.

where $nb(z)$ denotes the set of neighboring words of w_z and $p(w_i|w_z)$ denotes the hierarchical softmax of the associated word vectors $\mathbf{v}_{w_z}$ and $\mathbf{v}_{w_t}$ (see (Mikolov et al., 2013) for more details). Due to the simple architecture and the use of hierarchical softmax, the skip-gram model can be trained on billions of words per hour using a conventional desktop computer. It is unsupervised to learn the word embeddings and it can be computed on the corpus of interest or pre-trained in advance. Note our framework is not limited to a particular type of word embeddings, any commonly-used embeddings for text can be substituted into our framework.

3.2 Efficient simple attention

Attention mechanisms (Mnih et al., 2014) have been shown to be useful in deep learning models. They help guide pooling to be more cognizant of the key answer tokens relative to the question. While useful, using attention brings another layer of complexity, more model parameters, hence longer training time.

In this section, we ask the question – can we design a more efficient attention method in the fast model that does not require additional parameters yet achieves good results? The answer is in the affirmative. Our new attention strategy in the fast model is extremely simple (only 1 line of code!), however, it is surprisingly effective. For a pair of question and answer, we replace the question text by an augmented version of the question, which is formed by a simple *concatenation* of the original question and the given answer. This exact process is illustrated by the left side in Figure 1, where each w_m^q denotes a word in the question q, and w_n^a is a word in the answer a. The new question consists of the combined tokens from the question and answer. A look-up on word-embeddings is then performed to get the corresponding word vectors for the new question tokens. Then max- and min-pooling will be performed (details in the next section) on these word vectors. The final cosine score will be computed between the new question and the answer. Note on the answer side, the representation of the answer text stays intact (i.e., a lookup on word vectors of the original answer tokens is performed, and the vectors are passed to the pooling stage).

Why do we call this simple strategy an *attention* mechanism? Adding question q's tokens into the answer will "corrupt" the answer's representation and the subsequent max- and min-pooling results of the answer, by *biasing* the answer's representation towards q's semantic meaning. Intuitively, if the answer is correct for q, then they are semantically identical, in which case the "corruption" of the answer a by q should not change the answer's semantic meaning, and the subsequent cosine similarity between the answer a and the combined text should be very close to one (i.e., strong similarity). One the other hand, if the answer is incorrect for q, they are semantically different, the corruption of a by q will make the answer's representation drastically different from its original, forcing the cosine score with its original representation far from one.

The simple "corruption" by q can be viewed as an *attention* placed on the answer, by *focusing explicitly* on q's influence on the answer with respect to the answer's original representation. This is exactly what is captured by Figure 1. The new question can be viewed as a corrupted version of a (by adding q's tokens into it), and after the subsequent pooling operations, cosine similarities between the original answer's representation and the new corrupted version are computed.

3.3 Pooling and similarity computation

The next step in the fast model is the max- and min-pooling as shown in Figure 1. They are directly performed on word vectors from the answer and the modified question to form one-dimensional vector representations. The pooling is performed along each dimension in the word vector, over the tokens in the input text.

We can think of max-pooling as a way for the model to ask whether a given semantic class is found anywhere in the input text, while the min-pooling captures the absence of it. Both pooling techniques have been used before in deep learning applications. In this work, each pooling is used to create a pair of representations for the inputs, from which a cosine similarity score is derived. The resulting two cosine similarity scores (based on two pooling strategies) are then combined via a weighted linear combination to form a final score for the answer:

$$Score(a) = w \cdot cosine(v_{max}^{q'}, v_{max}^{a}) + (1 - w) \cdot cosine(v_{min}^{q'}, v_{min}^{a})$$

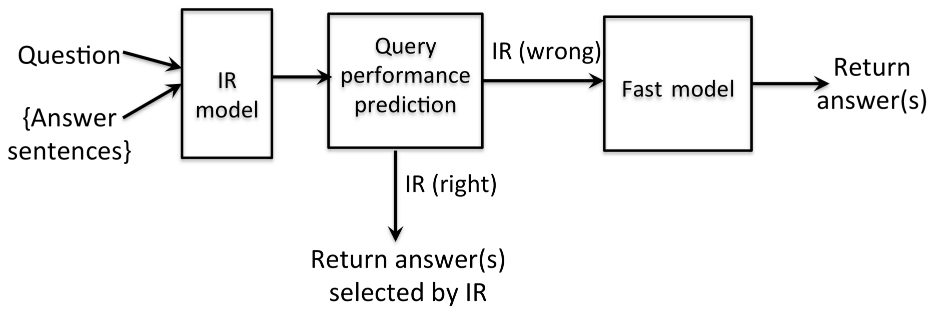


Figure 2: Hybrid approach combining IR with fast, simplified deep model (Figure 1)

where vectors $v^{q'}_{max}$ and v^{a}_{max} denote the outcome of max-pooling performed on the inputs (i.e., answer a and the augmented question q'). The vectors $v^{q'}_{min}$ and v^{a}_{min} are similarly defined. The single weight w in the linear combination is empirically set. We found in practice, this weight is fairly robust and insensitive to different datasets. In this work, it is simply set to a constant ($w = 0.7$) and does not change from datasets to datasets. Finally, $Score(a)$ is the score used to rank each candidate answer for the given question, and the one with the highest score is chosen as the best answer by this model.

3.4 Hybrid structure

A hybrid structure is used to combine the fast model with an IR retrieval model. In this work, we use the IR retrieval model in (Bendersky et al., 2010), which has shown dominant performance in a number of text and sentence retrieval tasks. The hybrid model is shown in Figure 2. It leverages complementary strengths of the IR model and the fast model. For accuracy, the IR retrieval model and the fast model each can cover a unique population of the questions. For example, the IR retrieval model excels at exact matching of the question with answer terms, if the answer contains sufficient word overlaps with the question. Predictably, a decent fraction of questions and answers may fall into this category, given the past wide usage and success of such retrieval models (Tao et al., 2007; Bendersky et al., 2010; Buttcher et al., 2006) for answer selection and retrieval. On the other hand, some correct answers may not share any words with the question, and in this situation, the fast model, which works from a semantic level, can be put into use.

A key component in the hybrid model is deciding when to return the IR results, and when to forward it to the fast model for scoring. This problem is formulated as "query performance prediction" (QPP), previously studied by the retrieval community (He et al., 2005; Hauff et al., 2009). A predictive model (usually of the form of a logistic regression) is used to predict the accuracy of the initial results for a given question, based on statistics extracted from IR answer scores for the question, such as the separation (ratio) between the maximum and minimum answer scores, etc. In this work, we follow the work by (He et al., 2005) to build the query performance prediction model, which is used to direct a question to either the fast model or return the IR results. It is beyond the scope of our paper to detail the QPP approach; we refer interested readers to (He et al., 2005) for more details about query performance prediction.

Finally, we would like to point out the entire pipeline is hyper-parameter free, relieving the need for expensive hyper-parameters tuning. This property allows our model to work with new domains and datasets more seamlessly – reducing the workload in model tuning.

4 Experiments

In this section we present a comprehensive set of experiments over three QA datasets: WikiQA, TrecQA, and InsuranceQA. WikiQA (Yang et al., 2015) is an open domain question-answering dataset. We use the subtask that assumes that there is at least one correct answer for a question. The TrecQA dataset was created based on TREC QA task (8-13) data (Voorhees et al., 2000). We follow the exact approach of train/dev/test question selections as in (Wang et al., 2015). InsuranceQA(v2)[2] is a recently released

[2]git clone https://github.com/shuzi/insuranceQA.git

	WikiQA	TrecQA	InsuranceQA(v2)
Train (# questions)	873	1162	12,889
Dev (# questions)	126	65	2000
Test (# questions)	243	68	2000
Avg # cand answers	9	38	500

Table 1: Dataset statistics: WikiQA, TrecQA, and InsuranceQA(v2)

large-scale non-factoid QA dataset from the insurance domain, which has drawn interests from the deep learning community for studying answer selections.

The statistics of these datasets are given in Table 1, including the number of questions, and the average number of candidate answers in the train/dev/test set. We study the following methods in our experiments:

- **IR retrieval model (Bendersky et al., 2010).** This IR model utilizes unigram and bigram overlaps between the question and answer, and represents the state-of-the-art in the IR field for effective answer selection and text retrieval (Bendersky et al., 2010). Similar to our approach, this model requires no training and it is applied the same way for all datasets.

- **FastHybrid**. The proposed hybrid model (Section 3) does not require any training. The tunable parameter (the weight w in the linear combination of cosine scores) is empirically set to 0.7 and stays the same for all datasets. Thus, the training and dev sets are not used by the hybrid model. In addition, the model uses standard word embedding vectors[3] trained from the Wikiepdia[4] and Gigaword [5] data collections. Although we do not re-train the word-embeddings and use the same pre-trained embeddings for all datasets, we could potentially re-train it on each dataset to achieve higher accuracy performance than reported here.

- **Supervised background models**. Since model scalability and efficiency are an issue, for supervised models, rather than re-training them for each new QA dataset, we could instead just train them on a big background dataset containing examples representative of the individual datasets, then apply the trained model to each QA data without re-training. This strategy refrains from repeated training, and scales better to large number of domains – thus, it serves as a direct comparison point to the hybrid model in the experiments. Here we use the Yahoo! answers dataset[6], which is a large Q&A corpus from which we extracted question-answer pairs as the training data. As the training method, we use two state-of-the-arts deep learning models (Feng et al., 2015; Santos et al., 2016), which apply convolutional neural networks to extract meaningful representations for each question and answer pair (Feng et al., 2015), and utilize bidirectional attentions on the questions and answers to enhance answer selection accuracy (Santos et al., 2016). We denote these two supervised background models as *Supervised background-1* and *Supervised background-2*, from applying (Feng et al., 2015) and (Santos et al., 2016) to the Yahoo background corpus, respectively.

- **Deep learning methods for answer selection**. As mentioned earlier, there is a recent surge on applying deep learning for answer selection. To be maximally effective, these techniques typically require in-domain training data for large-scale parameter tuning. Although our end goal is clearly different from that of standard in-domain supervised deep learning – we approach the problem from an accuracy *and* scalability point of view and develop light-weight models which require little tuning and adaptation across different domains – we present their results for completeness purpose whenever appropriate. As we will see, although the hybrid model uses no training data, its accuracy is often on-par (and sometimes slightly better) than supervised deep learning methods trained with in-domain data (Santos et al., 2016; Feng et al., 2015). This point echoes a similar observation made recently by related work on classification (Joulin et al., 2016).

[3]http://nlp.stanford.edu/projects/glove/
[4]https://dumps.wikimedia.org/enwiki/20140102/
[5]https://catalog.ldc.upenn.edu/LDC2011T07
[6]https://webscope.sandbox.yahoo.com/

Model	WikiQA top-1 acc	TrecQA top-1 acc	InsuranceQAv2 top-1 acc
IR (Bendersky et al., 2010)	40.9%	63.23%	18.20%
Supervised background-1	44.4%	61.7%	21.6%
Supervised background-2	44.4%	60.2%	19.4%
FastHybrid	48.2%	71.5%	22.7%

Table 2: Top-1 test accuracy on three QA datasets

Model	WikiQA training	TrecQA training	InsuranceQA(v2) training
IR model (Bendersky et al., 2010)	no training (0s)	no training (0s)	no training (0s)
Supervised background-1 model	6h/170K question-answer pairs		
Supervised background-2 model	6h/180K question-answer pairs		
FastHybrid	no training (0s)	no training (0s)	no training (0s)

Table 3: Training time on three QA datasets.

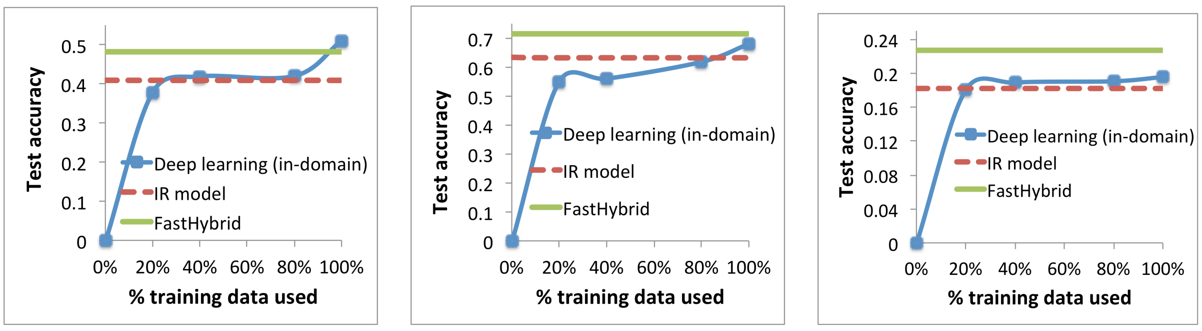
All experiments were run on a NVIDIA Tesla K20Xm GPU processor, with memory size per board (GDDR5) 5GB.

4.1 Model accuracy

Table 2 presents the top-1 accuracy for each QA domain test set[7]. Among the four comparison methods, IR and *FastHybrid* incur no training time (since no training is needed), while the supervised background models are trained on the large-scale background Yahoo Q&A corpus. We see that the hybrid model consistently achieves better accuracies for all QA domains in comparison to other techniques. An interesting observation is that while the hybrid model is simple, it is more robust than the supervised background models. For example, while in WikiQA and InsuranceQA, the supervised background models achieve decent performance relative to IR and *FastHybrid*, in TrecQA, their performance drops significantly (at 61.7%, and 60.2%, respectively) with respect to the hybrid model (71.5%). While utilizing the same training and tuning mechanism as the state-of-the-arts deep learning techniques (in terms of parameter tuning etc), it cannot make up for the domain gap between the TrecQA domain and the background Yahoo! Q&A corpus used as the training data. This points out that while the supervised background models can save some training time by doing a one-time offline training, it may potentially and significantly hurt the accuracy of new domains not well represented by the background corpus. This contrasts with the hybrid model, which aims to simultaneously achieve better efficiency, scalability, while not hurting accuracy.

Furthermore, while we try to build models that can scale to large number of domains more quickly and easily – a departure from standard in-domain supervised techniques (which can be used as an accuracy upper-bound), in many cases the hybrid model performs on-par with these methods for accuracy. As we will see in Section 4.3, the hybrid model (not using any training data) slightly outperforms supervised deep models (Feng et al., 2015) from in-domain training for two out of three datasets, while being significantly faster and more scalable – a highly desirable property for large-scale real-world applications. Note our original goal was to achieve no significant loss in accuracy while reducing costs. It is interesting

[7]Top-1 accuracy is one of the most commonly-used metrics to evaluate answer selection and question-answering. Other metrics include MRR (Radev et al., 2002) and MAP (Baeza-Yates et al., 1999). Since in this paper we focus on getting a correct answer rather than the entire ranking of answers, top-1 accuracy is reported.



(i) WikiQA (ii) TrecQA (iii) InsuranceQA(v2)

Figure 3: Top-1 test accuracy as the training dataset size is being varied from 0% to 100% of its original, in increments of 20%, for three QA datasets (i) WikiQA; (ii) TrecQA; and (iii) InsuranceQA(v2).

to see that the hybrid can surpass the expectation, and not only performs well for time, but for accuracy equally. This confirms our earlier observation that the simple attention mechanism used by the model works quite well in practice.

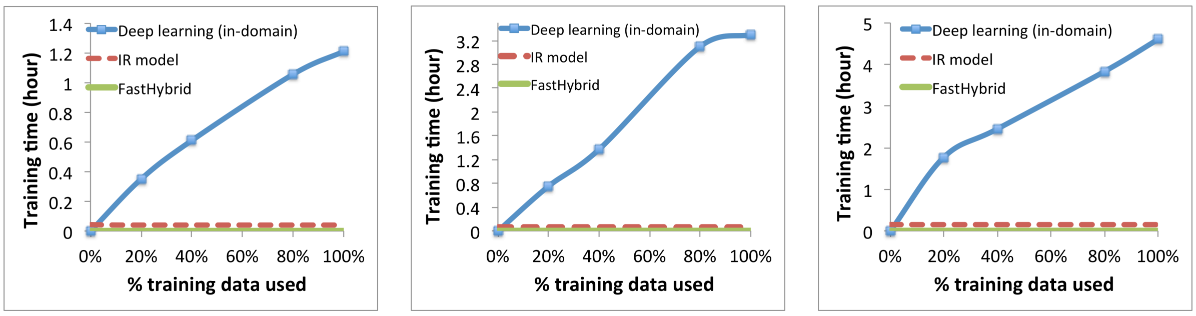
4.2 Training time

Table 3 presents the training time for each model. As mentioned earlier, all training and test experiments were carried out on a NVIDIA Tesla K20Xm GPU. As expected, both the IR model and *FastHybrid* are highly efficient. The supervised background models are more expensive, since they have to be trained on a large-scale background corpus. We note the training times of the supervised background models depend on the values of their hyper-parameters (e.g., # convolution filters, context window size etc). They are set in accordance to the settings used by the authors of these deep learning models. We also note that the supervised background models only have to be trained once. This is significant when we want to scale to a large number new domains (e.g., in the order of thousands). The saving in time across these many domains can add up to be quite noticeable. Nonetheless, when taking both accuracy and scalability/efficiency into account, we would like to have a model that has a higher accuracy yet not incurring too much costs (in training, tuning etc.), which is achieved by the hybrid model.

4.3 Test accuracy vs training set size

Next, we would like to ask the question – assume we follow the standard supervised setup where in-domain training data is used to train a model each time, can we improve deep learning model training efficiency by using a reduced set of the data and achieve similar test accuracy as using the full training set? Figure 3 and Figure 4 explore the effects of in-domain training set size on test accuracy and training time, respectively. Figure 3 shows the test accuracy as a function of the amount of training data used, and each reduced training set is a random sample from the full training data (i.e., sampled at 20%, . . . , 80% of the full set). Figure 4 reports the corresponding training efficiency achieved at each training set size. Note it is clear for IR and the *FastHybrid*, they reside along the x-axis in Figure 4 which denotes their constant (0) training time.

Given the results shown earlier (Table 2) where the deep learning training method (Feng et al., 2015), when applied to the Yahoo background corpus (supervised background-1), slightly outperforms supervised background-2, we employ (Feng et al., 2015) for in-domain training, denoted by "Deep learning (in-domain)" in Figure 3 and Figure 4. Note at 100% training data, this represents a standard in-domain supervised deep learning model. As we can see from Figure 3, the hybrid model achieves equal/better accuracies for two out of three datasets (TrecQA and InsuranceQA), as compared to the supervised in-domain deep learning model from using *full* training data. Furthermore, for the deep learning model, it is clear that reduced training sets lead to much improved training efficiency. For example, at 20% of the original training set size (Figure 4), its training time drops to 0.35h, 0.75h, and 1.77h for WikiQA, TrecQA, and InsuranceQA, respectively – a decent improvement over its original time (1.21h, 3.3h, 4.6h, respectively). However, the enhanced training efficiency comes at a cost of reduced test accuracy. As



(i) WikiQA (ii) TrecQA (iii) InsuranceQA(v2)

Figure 4: Training time as the training dataset size is being varied from 0% to 100% of its original, in increments of 20%, for three QA datasets (i) WikiQA; (ii) TrecQA; and (iii) InsuranceQA(v2).

shown by Figure 3, when no training data is used, its test accuracy drops to 0%. In the future, it would be interesting to look into how to deal with very few training data for these techniques, a problem that has been drawing much interest recently (Socher et al., 2013; Romera-Paredes et al., 2015; Palatucci et al., 2014; Ba et al., 2015).

5 Conclusion

In this work, we have developed the *FastHybrid* model, a hybrid model which combines a fast model with an IR model to form an efficient and effective hybrid structure for the answer selection task. Unlike the previous deep learning models for this task, our hybrid model is nearly hyper-parameter/parameter-free, so it is extremely efficient (no training) for different datasets, and as a result, it can scale very easily to a large number of new domains and users. We performed a set of extensive experimental studies that demonstrate both the accuracy and training efficiency of our new method, as compared to several strong baselines noted for their accuracy and efficiency. In the future, we plan to explore applying this model to related tasks such as recommendation. In addition, we are interested in building and plugging in additional query performance prediction (QPP) models into our hybrid model.

6 Acknowledgements

We sincerely thank Bowen Zhou and Bing Xiang for comments that improved the manuscript, and Cicero dos Santos for assistance with the deep learning software used in this paper.

References

Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting Deep Zero-Shot Convolutional Neural Networks using Textual Descriptions. In *IEEE International Conference on Computer Vision.*

Ricardo Baeza-Yates, and Berthier Ribeiro-Neto 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc.

Michael Bendersky, Donald Metzler, and Bruce Croft. 2010. Learning Concept Importance Using a Weighted Dependence Model. In *ACM International Conference on Web Search and Data Mining.*

Stefan Buttcher, Charles Clarke, and Brad Lushman. 2006. Term proximity scoring for ad-hoc retrieval on very large text collections. In *The 29th annual international ACM SIGIR conference on Research and development in information retrieval.*

Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. 2014. Building an efficient and scalable deep learning training system. In *The 11th USENIX conference on Operating Systems Design and Implementation.*

Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. 2013. DLID: Deep Learning for Domain Adaptation by Interpolating between Domains. In *ICML 2013 Workshop on Representation Learning.*

Oren Etzioni. 2011. Search needs a shake-up. In *Nature. 476(7358):25–26*

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: a study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU).*

David Ferrucci. 2012. Introduction to “This is Watson”. In *IBM Journal of Research and Development. 56(3.4).*

Yaroslav Ganin, and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning.*

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *International Conference on Machine Learning.*

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing.*

Claudia Hauff, Leif Azzopardi, and Djoerd Hiemstra. 2009. The Combination and Evaluation of Query Performance Prediction Methods. In *European Conference on Information Retrieval.*

Ben He, and Iadh Ounis. 2005. Query Performance Prediction. In *Information Systems, 31(7).*

Sepp Hochreiter, and Jurgen Schmidhuber. 1997. Long short-term memory. In *Neural Computation.*

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *http://arxiv.org/pdf/1607.01759.pdf.*

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at ICLR.*

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuogl. 2014. Recurrent Models of Visual Attention. In *Annual Conference on Neural Information Processing Systems.*

Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot Learning with Semantic Output Codes. In *Annual Conference on Neural Information Processing Systems.*

Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating Web-based Question Answering Systems. In *The International Conference on Language Resources and Evaluation.*

Bernardino Romera-Paredes, and Philip H. S. Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning.*

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. *http://arxiv.org/pdf/1602.03609.pdf.*

Aliaksei Severyn, and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Richard Socher, Milind Ganjoo, Christopher Manning, and Andrew Ng. 2013. Zero-Shot Learning Through Cross-Modal Transfer. In *Annual Conference on Neural Information Processing Systems.*

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).*

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.*

Tao Tao, and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *The 30th annual international ACM SIGIR conference on Research and development in information retrieval.*

Ellen Voorhees. 2000. Overview of the Trec-9 Question Answering Track. In *TREC conference.*

Di Wang, and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Eric P. Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. 2015. Petuum: A new platform for distributed machine learning on big data. In *CM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *The 51st Annual Meeting of the Association for Computational Linguistics*.

Lei Yu, Karl Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Deep Learning Workshop*.

Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. 2012. Online Incremental Feature Learning with Denoising Autoencoders. In *The 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*.