Duplicate Question Pair Detection with Ensemble of Feature Engineering and Deep Neural Networks

Haoran Shi
haoransh@andrew.cmu.edu
Carnegie Mellon University

Weitian Ding
weitiand@andrew.cmu.edu
Carnegie Mellon University

Wenyan Hu
wenyanhu@andrew.cmu.edu
Carnegie Mellon University

Abstract

Recently, duplicate question detection brings about much attention with the rising need to maintain canonical question list of online question-answering platforms. We implement a diverse set of state-of-art deep learning models for duplicate question pairs detection, and we propose a novel architecture for duplicate question detection that ensembles deep learning models and hand-crafted features. Our best single model, BERT, achieves 89% accuracy and ensemble model achieves 90% testing accuracy on Quora Question Pair dataset, which outperforms most other neural network models and traditional paraphrase detection methods.

1 Introduction

For the convenience of management and browsing, it is important for those question-answering websites to detect and remove duplicate questions. Multiple questions with the same intent could cause writers writing multiple versions of the same answer and readers wasting their time seeking the best answer. To encourage the research community to solve this problem, Quora released the first large-scale duplicate question detection task, which can be viewed as a binary sentence classification task. Formally, given two questions q_1 and q_2, we need to decide the function

$$f : (q_1, q_2) \rightarrow \{0, 1\} \tag{1}$$

Identifying duplicate Quora questions can be viewed as an instance of paraphrase detection in a more restrictive context, which is a very important task in Natural Language Processing and has a wide application in information retrieval, question answering, text summarization, plagiarism detection and evaluation of machine translation [1]. Other than the common challenge encountered by all paraphrase detection task that there are many different sentences could convey the same meaning, what makes this dataset especially difficult is that two questions could be asking for different things but could be addressed with the same solution, which makes the two questions semantically equivalent.

2 Related Work

Semantic equivalence is a long-standing task in natural language processing, which is well-known to be difficult due to the complex expression methods in natural language. The detection of semantic equivalent questions raised a lot of attention as the question answering community sites become popular these years[2]. As the volumn of the questions becomes much larger, the management of these sites requires to remove or merge duplicate questions from the database if possible, to reduce the

Preprint. Work in progress.

redundancy overhead. However, this is not an easy task since the same question can be formulated in distinct ways, especially when the knowledge background of different users is not aligned. Bogdanova et al. (2015) defined two questions as semantically equivalent if they can be adequately answered by the exact same answer, and proposed a classification model with convolutional neural network for this task[3]. The release of Quora dataset on Kaggle facilitates the evaluation of this task and witnessed a lot of progress.

Most existing work to detect question paraphrase can be divided into two categories: traditional feature engineering methods and deep learning models. For the feature engineering, commonly used hand-crafted features consist of term co-occurrence statistics like tf-idf and n-gram overlap of characters/words, and similarity/distance in lexical database. In addition to these hand-crafted features, the neural models are proven effective in extracting hidden features from the plain text for the downstream applications[4, 5]. There are a diverse array of neural-network-based approaches that we can borrow for duplicate questions pairs detection [6, 7, 8]. Especially, the single-layer perceptron model with Pretrained BERT as the sentence encoder achieved the first place in General Language Understanding Evaluation(GLUE) benchmark, which is a collection of diverse natural language understanding tasks including the Quora Question pairs (QQP). Inspired by this, we would like to try pre-trained BERT along with other neural network based approaches in our project and explore whether we could achieve better performance with feature fusion and model ensemble.

3 Methodology

In the following section, we will introduce our data set preprocessing, the four basic single models and their experimental settings respectively; and our ensemble model which is based on hand-crafted features and the predictions of best-performing neural networks.

3.1 Dataset

The Quora Question Pair dataset contains 404,351 potential duplicate question pairs. Each sample contains the IDs for each question in the pair, the full text of each question, and a binary label indicating whether the line truly contains a duplicate pair. There are 255,045 sentence pairs with label False, which is 63% of the dataset. Some examples from the dataset are shown in Table 1.

id	question1	question2	is_duplicate
0	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
7	How can I be a good geologist?	What should I do to be a great geologist?	1
20	Why do rockets look white?	Why are rockets and boosters painted white?	1

Table 1: Examples in Quora Question Pair Dataset

Simple data analysis shows that medium question length is 11 words, the maximum is 272 words. 98% of the questions have less than 30 words. The most common words in the dataset are stop words such as "what", "the" and "is". 0.17% of the question pairs contain math formulas. If we use GloVe 6B common crawl as our vocabulary, then 90% of the questions have less than or equal to 1 out of vocabulary words (OOV) [9].

We also observe that, even if there is much word overlap between two sentences, it does not mean they are duplicate questions. This indicates that simple word matching statistics is not sufficient for this task. This is also one motivation why we want to try neural models to extract hidden features. In the experiments, we split the complete dataset into training, validation and test dataset randomly with a ratio of 7:1:2. In this way, we obtain 283,045 training examples, 40,435 validation examples, and 80,781 test examples.

We also try to expand our training set with Microsoft Research Paraphrase Corpus (MRPC), which contains 5,800 sentence pairs hand-labeled with a binary judgment as to whether the pair is semantic equivalent[10]. But we do not observe any performance increase with this additional training corpus, so we do not use this dataset in the following experiments. We conjecture that it is due to the distribution of MRPC may be different from that of Quora dataset, and the training over this dataset may degrade the performance. And due to the much smaller volume of MRPC than Quora dataset, the effect to the accuracy is negligible.

3.2 SVM Baseline with Hand-crafted features

Though it has been practically proved that neural network could capture some latent variables that are not usually captured by traditional hand-crafted linguistic features, we want to experiment with classic handcrafted features first to give us more insight into different hand-crafted features and this data set. We also want to use these features in our ensemble models.

We try an SVM classification model using the features as follows. (i) Basic features including: Word count of the first sentence, word count of the second sentence, word count difference; (ii) Term cooccurrence including: Word overlap; Character unigram overlap; Character bigram overlap; Character trigram overlap; Character 4-gram overlap; (iii) Linguistic Metrics which captures lexical similarity including: Unigram-BLEU, Bigram-BLEU and their weighted average; (iv) String metrics including: Jaccard similarity, Hamming distance and Levenshtein distance; (v) Tf-idf cosine similarity of sentence pairs. We discuss them in details in the followings,

String Metric String metrics simply measures the distance of two questions without considering the underlying sentence structure or their semantic meanings.

BLEU score BLEU score [11] is frequently used in machine translation to evaluate the similarity between machine translated sentence and reference sentences translated by human experts. BLEU score works by counting matching n-grams in the candidate sentence to n-grams in the reference sentence; it differs from simply counting the number of overlap n-gram method by considering the repetitive n-grams. We used individual unigram and bigram BLEU scores and cumulative BLEU-2 score with weights $0.5, 0.5$.

tf-idf Cosine Similarity The cosine similarity between the tf-idf encoded vectors of the two sentences is a very classic measure of document similarity in information retrieval. We choose not to filter out the stop words as most of other information retrieval tasks do, as the particularity of this dataset, some common stop words like "what", "which", "who" could be informative and crucial to determine the semantic equivalence of sentence pairs. For the same reason, we do not use stemmer to process the sentence to preserve affixes of words.

Part of Speech and WordNet Synset-based Semantic Similarity However, with all the above-mentioned metrics, we could only capture the lexical similarity of sentence pairs. Consider two questions: "How much is a bottle of alcoholic drink?" and "Average price of red wine". Although they share no common words, they can be effectively answered by similar answers. To solve this problem caused by the use of synonyms, we resort to WordNet, a lexical database which groups set of synonyms called synsets. Adapted from Pawar's work [12], we implement the following approach to calculate semantic similarity,

1. Firstly, we label words with their part-of-speech tag for disambiguation
2. Then, we associate each word with their corresponding WordNet synset
3. Word semantic similarity score is calculated by the shortest path distance between their synsets given that those two words have the same POS tag respectively.
4. A semantic vector is computed for each sentence and the length of the semantic vector is the same as the size of union of two word sets: each element is the semantic similarity score obtained by the word in the sentence which is most similar to the corresponding word in union of the word sets.
5. The final semantic similarity of two sentences is calculated using the cosine similarity of their semantic vectors

For example, the semantic similarity score of the above mentioned example is 0.01 though there is no common word shared by them. However, due to the speed of generating semantic similarity score (it takes on average 2-3 seconds per sentence pair in our experimental test on an Intel i5 workstation), we do not include this feature into the SVM classifier.

3.3 Bi-GRU Siamese Network

We discuss 4 neural-network based approaches we implement in the following sections. Siamese architecture is widely adopted in information retrieval and document similarity analysis where the training data can be structured into a pairwise format [6]. We implement a Siamese architecture using bi-directional GRU network (bi-GRU) for duplicate Quora questions detection. For each question, we use NLTK to tokenize Quora questions into words and truncate the word sequences to a maximum length of 32 [13]. Each word in a sequence is represented using 300-dimension GloVe and the sequence of word vectors are fed into a bi-GRU network to obtain a question vector [9, 14, 15]. The GRU network has 128 hidden units in each direction and the two GRU networks used to encode two questions share weights. Finally, The two question vectors are concatenated and fed into a fully-connected network which outputs the probability that the two questions are duplicates. The fully connected has two layers; the hidden layer has 64 tanh-activated units and the output layer has 1 sigmoid-activated unit. We supervise our bi-GRU Siamese network using log-loss and train with Adam optimizer with 0.001 learning rate. The mini-batch size for this network is 512. We also apply batch normalization on hidden-unit activation to regularize the network [16].

In the Siamese architecture, the two input questions are encoded into vector representations in an embedding space which captures the high-level semantic meaning of questions. However, fine-grained information which are important to our task may be lost in this process. In the following sections, we implement neural networks that model fine-grained interactions between two questions.

3.4 Match Tensor

Match Tensor models the interaction of topicality signals between two questions using a 3D tensor in which one dimension corresponds to words in the first question, one dimension corresponds to words in the second question, and the third dimension corresponds to different match channels [7]. Since the Match Tensor is applied on document retrieval, we implement a modified version of the original architecture for duplicate question detection. We use the same tokenization and word embedding process as in Section. 3.3 to obtain sequences of word vectors. Then, we encode both questions into sequences of hidden states using bi-GRU networks with 128 hidden units in each direction. The two bi-GRU networks share weights. We thus obtain 32×256 representations, Q_1 and Q_2, for the two questions, where 32 is the temporal dimension, and 256 is the bi-GRU hidden state dimension. Let $Q_{1,i}$ be the i-th column of Q_1, and $Q_{2,i}$ the i-th column of Q_2. Then the i-th match channel of Match Tensor M is obtained by

$$M_i = Q_{1,i} Q_{2,i}^T \tag{2}$$

Thus M has dimension $32 \times 32 \times 256$. Next we feed M into a convolutional neural network with 3 convolution layers. Each convlution layer has 128 3×3 filters with 1×1 strides. The output of convolution is relu-activated, and then pooled using a max-pooling layer of pool size 2×2 and stride size 2×2. We obtain a $4 \times 4 \times 128$ tensor D as the output of the convolutional network. Finally, we use global max pooling on each channel of D and obtain a 128 dimensional vector d, and we feed d into a fully conncected network with 64 tanh-activated hidden units and 1 sigmoid-activated output unit which represents the probability of duplicate questions. We use log-loss and Adam optimizer with learning rate 0.001 to train our Match Tensor network. The mini-batch size is 1024. We batch normalize the activation of convolution layers and fully connected layers to regularize the network.

3.5 Co-attention Network

Several sate-of-art deep learning model for natural language processing incorporates an attention mechanism. Xiong et al. propos a co-attention network that attends to document and questions simultaneously for a question answer task [8]. We propose a novel architecture in detecting duplicate questions borrowing the co-attention mechanism. We first obtain sequences of hidden states, Q_1 and Q_2, for both questions using the same bi-GRU network as in section 3.4. Q_1 and Q_2 are 32×256 matrices, where 32 is the temporal dimension and 256 is the hidden state dimension. We compute an affinity matrix $A = Q_1 Q_2^T$, and we scale A to $[0, 1]$ by row to obtain A_1, and by column to obtain A_2. And thus the summaries or attention contexts for question 1 in light of question 2 is $C_1 = A_1 Q_1$. Similarly, the summaries of question 2 in light of question 1 is $C_2 = A_2^T Q_2$. Next, we concatenate C_1 and C_2 and feed it to another bi-GRU network with 128 hidden units in each direction. The output of the bi-GRU network is fed into a 2-layer fully connected network which outputs the probability of duplicate questions. We use log-loss and Adam optimizer with learning rate of 0.001 to train the co-attention network. We use mini-batch size of 1024. We use batch normalization to regularize the last hidden layer.

3.6 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a recently proposed language representation model, which achieved the first place in Gluebenchmark recently[5, 17]. Inspired by the idea that deep bidirectional model should be more powerful than either a left-to-rigth model or the shallow concatenation of left-to-right and right-to-left model, the authors propose to pretrain the Transformer Encoder with bidirectional self-attention. Pretrianing is performed on two tasks: 'masked language model' to capture the conditional distribution of words in one sentence, and 'next sentence prediction' to capture the relation of different sentences.

We adopt the pretrained model with 12 layer submodules, 768 hidden dimension and 12 attention heads in our experiment. Also, following the official configuration of BERT, we use WordPiece tokenization, add the special token '[CLS]' indicating the beginning of a sentence, concatenate two questions in one pair with the special token '[SEP]'. For the classification task, we use one linear hidden layer with 768 dimension and linear output layer with 2 dimension, and use binary cross entropy loss and Adam Optimizer with weight decaying and maximum learning rate 2×10^{-5} to train the complete model including the pretrained BERT. The training batch is 32, and the max sequence length after concatenation is 128. We also adopt the early-stopping to prevent overfitting based on the performance on the validation set[18]. We re-implement the complete model with Texar library and our showcase has been merged into its official repository[19].

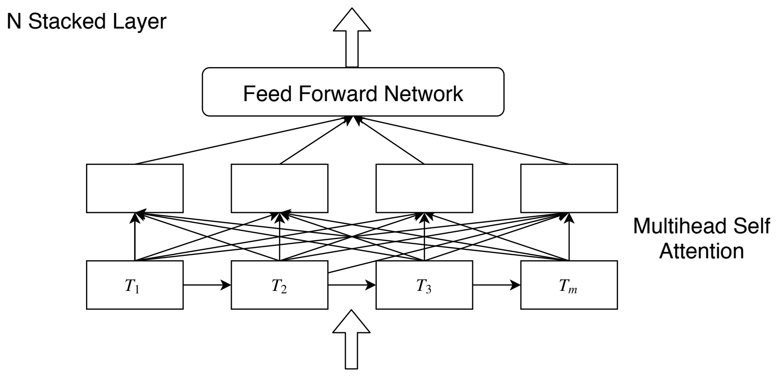

Figure 1: Model architecture of BERT

In view of the BERT is a pretrained model of Transformer Encoder, we also train the Transformer Encoder for sentence pair classification from scratch. We follow the original base transformer model from Vaswani *et al.* (2017), with 6 layer submodules, 512 hidden dimension and 8 attention heads as the substitution of BERT[20]. We use 128 training batch size and Adam Optimizer with maximum learning rate 1.0×10^{-3}. For more details, you can refer to our Github codebase [1].

3.7 Ensemble

We experiment with different combination of single models, combining with hand-crafted features to build our ensemble model. We explore two strategies for the ensemble models. First, we use the activations in the last hidden layers of our 4 neural networks with hand-crafted features as input, and use Random Forest and XGBoost as the final ensemble models respectively (Figure 2). We expect to combine the strength of different neural networks with traditional feature-engineering methods to achieve better performance than single models. But due to high hidden-layer dimension ($\sim$1000), these ensemble algorithms are too slow to converge; the last hidden layer of BERT has 768 units, and the other 3 neural networks have 64 units. Then, we resort to using the predictions of neural networks as input to ensemble models, instead of the activations of the last hidden layers.

[1]https://github.com/haoransh/Duplicate_Question_Detection

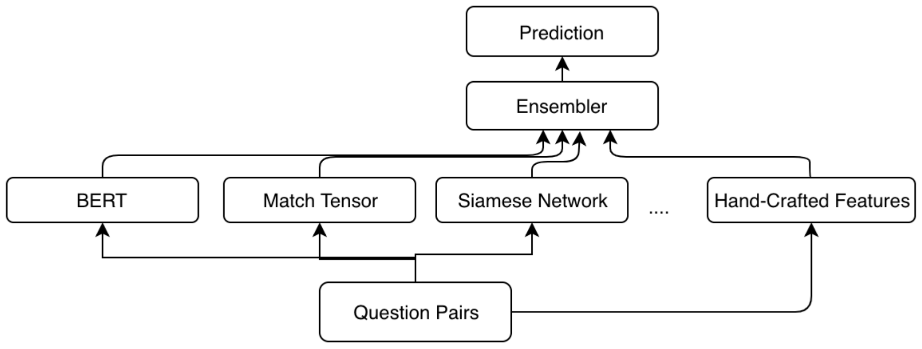

Figure 2: Ensemble Architecture

In our experiments, if we train the ensemble model with the same training set of the neural networks, only using the output of BERT can achieve better ensemble results than using combinations of other neural networks introduced above. The reason is that other networks have more parameters and they are trained with more epochs on the training set, so they overfit the training set more significantly than the pre-trained BERT, which leads to poor generalization ability on the testing set. So we train our ensemble models on the validation set of the neural networks instead.

Specifically, in our ensemble models, we use the hand-crafted features introduced in the feature fusion network, which proves to be the most stable and efficient[2]. For the Random Forest, we use 100 estimators and set the max tree depth as 5. For XGBoost, we use 100 estimators or decision trees, and we sub-sample 80% of the training data to train each estimator to increase generalization ability. We do not sub-sample features in XGBoost.

4 Experiments and Results

Our experimental results are shown in Table 2. Notice that even if we use early-stopping to prevent overfitting in all neural network models, all of the machine learning models seem overfitting in the training set.

Among all the single models, we could observe BERT achieves the best accuracy in the test set. The performance gap between BERT and Transformer from scratch demonstrates the efficacy of pretrained language representation in BERT. Also, BERT achieves the best performance with only $9k$ iteration steps. Recall that tuned batch size of BERT is 32, which means BERT can be trained well with only 1 pass over the training dataset. On the contrary, if we train the transformer model from scratch with a smaller network configuration than BERT, the model converges in 10 epochs, and the performance is not comparable to BERT. Our experiments demonstrate the superior generalization ability of fine-tuning BERT in this task. We observe other neural network based single models we implement have comparable performance and saturate at 0.83 test accuracy. These networks are much faster train than BERT but overfit after 4 to 5 epochs. One possible explanation is that the training data size is too small compared to the number of trainable parameters in these networks. This also explains why pre-trained BERT significantly outperforms training BERT from scratch.

For the ensemble model, XGBoost achieves best testing accuracy. We show feature importance in Figure 3. The predictions from 4 neural networks are the 4 most important features. Not surprisingly, the relative feature importance of neural network predictions is consistent with their single model performance. However, our ensemble models do show better generalization than the single models.

4.1 Error Analysis

The ground-truth labelling is not free of error in this dataset. For example, we have encountered two exact same questions being labelled as not duplicate questions. Therefore, it is difficult to

[2]https://github.com/hengluchang/Quora-Paraphrase-Question-Identification

Model	Train Accuracy	Val Accuracy	Test Accuracy	Test F1
SVM Baseline	0.756	0.7242	0.7248	0.6380
Transformer from scratch	0.8301	0.7606	0.7569	0.6506
Co-attention	0.8538	0.8093	0.8066	0.7332
Siamese Network	0.8860	0.8185	0.8158	0.7407
Match Tensor	0.9082	0.8362	0.8337	0.7838
BERT	0.9149	0.8920	**0.8896**	**0.8556**
Random Forest Ensemble	-	-	0.8915	0.8562
XGBoost Ensemble	-	-	**0.8955**	**0.8606**

Table 2: Performance of Different models. Top list the single models, The bottom list the ensemble models, which using hand-crafted features and the predictions from single models with bold text as input. Since we train ensemble models on the validation set, we do not show their train accuracy and validation accuracy.

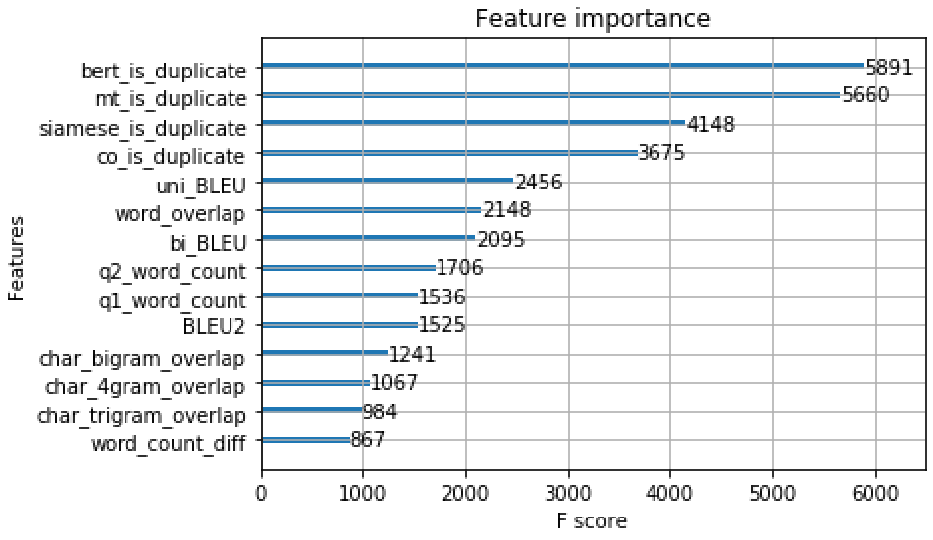

Figure 3: XGBoost Feature Importance. {bert, mt, siamese, co}_is_duplicate represents the predictions from BERT, Match-Tensor, bi-GRU siamese, and co-attention networks respectively.

quantitatively analyze the percentage of each type of error. We sample several erroneously classified question pairs and found the majority of errors can be categorized into the following types, which are shown in Table 4.1. We can observe the true labels of these cases are ndeed nontrivial and not easy to recognize even for humans.

5 Conclusion

In this project, we implement a diverse set of handcrafted features and neural-network based models for identifying semantic equivalent questions. We propose a method to fuse of our best performing neural networks and hand-crafted features. From our experiments, we find that BERT achieves 89% testing accuracy, significantly outperforming all other single models. We also show that our ensemble model achieves 90% accuracy, outperforming the best single model by a small margin.

Future Work Our experiments show that the pre-trained BERT outperforms other neural network based models including training BERT from scratch. This suggests that our dataset may be inadequate for training large neural networks. One promising direction is to incorporate external dataset or augmenting the existing dataset to obtain more training samples. Also, considering our ensemble model with traditional features and deep learning models achieves slightly better performance, we think it may be helpful to add more meaningful linguistic features in our model, e.g., the semantic

Error Reason	Sample Question Pairs	Ground Truth	Our Prediction Result
Real World Knowledge	1)What is it like to live in Cologne? 2) What is it like to live in Köln, Germany?	True	False
Numerical Reasoning	1) How can I increase my height after 21 also? 2) Can height increase after 25?	True	False
Words Overlap	1) Will there be another billion dollar lottery Jackpot? 2) When will the lottery reach the next billion dollar jackpot?	False	True
Rhetorical Figure	1) What is the reason Pakistan supports terrorism? 2) Has Pakistan become a safe haven for global terrorism?	True	False
Other	1) How do most people die? 2) How do people die?	False	True

Table 3: Error Samples

similarity calculation based on an external graph like WordNet Synset. We believe this should effectively resolve the error caused by lacking real-world knowledge mentioned above.

References

[1] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809, 2011.

[2] Yun Zhang, David Lo, Xin Xia, and Jian-Ling Sun. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology*, 30(5):981–997, 2015.

[3] Dasha Bogdanova, Cicero dos Santos, Luciano Barbosa, and Bianca Zadrozny. Detecting semantically equivalent questions in online user forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 123–131, 2015.

[4] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792, 2016.

[7] Aaron Jaech, Hetunandan Kamisetty, Eric Ringger, and Charlie Clarke. Match-tensor: a deep relevance model for search. *arXiv preprint arXiv:1701.07795*, 2017.

[8] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[10] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[12] Atish Pawar and Vijay Mago. Calculating the similarity between words and sentences using a lexical database and corpus statistics. *arXiv preprint arXiv:1802.05667*, 2018.

[13] Steven Bird and Edward Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.

[14] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[17] Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[18] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[19] Zhiting Hu, Haoran Shi, Zichao Yang, Bowen Tan, Tiancheng Zhao, Junxian He, Wentao Wang, Xingjiang Yu, Lianhui Qin, Di Wang, et al. Texar: A modularized, versatile, and extensible toolkit for text generation. *arXiv preprint arXiv:1809.00794*, 2018.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.