Duplicate Question Detection - A Semantic Learning Approach

Anvay Govind Pandit
Texas A&M University
College Station, Texas

Sarwesh Krishnan
Texas A&M University
College Station, Texas

Shibin Tazhe Veettil
Texas A&M University
College Station, Texas

Harinath
Texas A&M University
College Station, Texas
vharinath1105@tamu.edu

1 ABSTRACT

The problem of identifying duplicate questions is particularly significant in question answering platforms like Quora, Reddit, StackOverflow etc. because of the high quantum of users accessing these websites. Through this project we built a classifier capable of identifying duplicate questions given a question pair as input. In addition, we also hope to deploy the model as an application while understanding the system constraints existing in the real world. First, we began by creating Machine Learning models(SVM and XgBoost) with handcrafted features. We utilized the insight we gained from these approaches while building the Deep Learning models. We achieved a classification accuracy of 86.2 % by combining a BiLSTM model with handcrafted features. However, we were able reach 91.6% on the validation set by employing pretrained BERT representations. Finally, we built a practical system for deploying our classifiers.

KEYWORDS

Machine Learning, LSTM, BiLSTM, XgBoost, SVM, glove, word2vec, TF-IDF, BERT

2 INTRODUCTION

Asking and answering are essential components for learning. Today, there exists numerous online platforms which facilitate this experience. The number of users accessing these websites have grown significantly in the recent years. Particularly, Quora alone has reached 300 million monthly users.

Unfortunately, these portals are plagued by "duplicate" questions which creates a bad user experience. We define a duplicate question as a restatement of the first question using a different set of words. Duplicate questions can cause seekers to spend more time finding the best answers, while making the writers answer multiple versions of the same question. In this project, we try various approaches to automatically detect duplicate questions in order to establish a single source of truth for similar questions.

2.1 Problem Background

In this project we utilized the Quora Question Pairs dataset which was released as part of a Kaggle contest in March, 2017. The goal of that competition was to predict which of the provided pairs of questions were duplicates. Back then, Quora was using a Random Forest model to tackle this problem.

3 RELATED WORK

A simple duplicate detection approach is a word-based comparison. For example, we can use standard information retrieval measures like TF-IDF or BM25 to find the word-based similarity between two questions, and then classify question pairs with similarity scores below a threshold as duplicates. But, if we want to detect duplicates at a more intent-based, semantic level, we need intelligent approaches. In fact, there is active ongoing research in the Natural Language Processing community on this very problem [1-3].

Quora's current production model [6] for solving this problem uses a Random Forest based on numerous handcrafted features, including the cosine similarity of the average of the word2vec [7] embeddings, the number of common words, the number of common

topics labeled on the questions, and the part-of-speech tags of the words.

Because of their superior ability to preserve sequence information over time, Long Short-Term Memory (LSTM) networks [10], a type of Recurrent Neural Network, have obtained strong results on a variety of sequence modeling tasks. The LSTM based approach for duplicate question detection [1] achieved 79.5% F1 with 83.8% accuracy on the test set. Bi directional LSTMs [9] have been widely used for various NLP tasks including duplicate question detection since it takes into account both current and future state while processing each words.

BERT (Bidirectional Encoder Representations from Transformers) [2] has started a revolution in NLP with state-of-the-art results in various tasks. BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks without substantial task-specific architecture modifications. It obtained new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE benchmark to 80.4% , MultiNLI accuracy to 86.7 etc.

4 DATASET

As mentioned, we used the "Quora Question Pairs" dataset. There are a total of 404,351 question pairs in this dataset. Each row has the following structure:

- Index - The entry number
- Qid1 - The technical id of the first question
- Qid2 - The technical id of the second question
- Question1- The textual representation of the first question
- Question2- The textual representation of the second question
- Is_duplicate - A binary label where 1 indicates that the questions are duplicate and 0 indicating otherwise

5 METHODOLOGY

We tried multiple approaches for this problem. In addition to performing feature engineering and using traditional ML models like XgBoost we also implemented deep neural networks. To increase the classification accuracy, we further combined the representations of a particular Bi directional LSTM model with a few statistical features. However, we achieved the best results while using the pretrained BERT model.

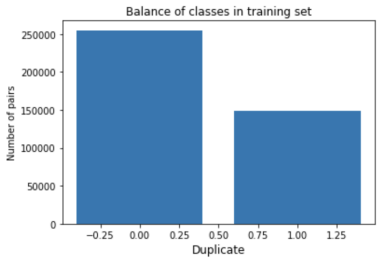

Figure 1: Distribution of target data. 63% of duplicate question pairs.

5.1 Exploratory Data Analysis

(1) *Loading the data*: The data is loaded into a pandas dataframe with nearly 400K rows in train.csv.
(2) *Missing data*: Only Question 2 has 2 missing values which are filled with blank.
(3) *Distribution of target data*: There are more number of unique question pairs (63%) as compared to duplicate question pairs. You can see the same in *figure 1*.
(4) *Unique questions*: Total number of Unique Questions are: 537933. Number of unique questions that appear more than one time: 111780 (20.8%). Hence, nearly 80% of questions are unique. Max number of times a single question is repeated: 157. *Figure 2* shows the same.
(5) *Feature Engineering and their plots* : For feature engineering we are going to look at several features: Length of each of question in words and chars; Share of words in each question pair; Similarity Scores - Jaccard, overlap, ochiai/cosine(If we treat sets as bit vector) for average; Cosine distance using vector representation We normalize features to a standard scale between 0 - 1 using min max scaler. These features are plotted as scatter plots in *figure 3*.

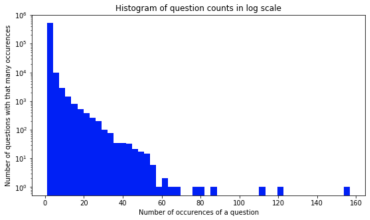

Figure 2: Majority of the questions have just 1 occurence and very few questions appear multiple times.

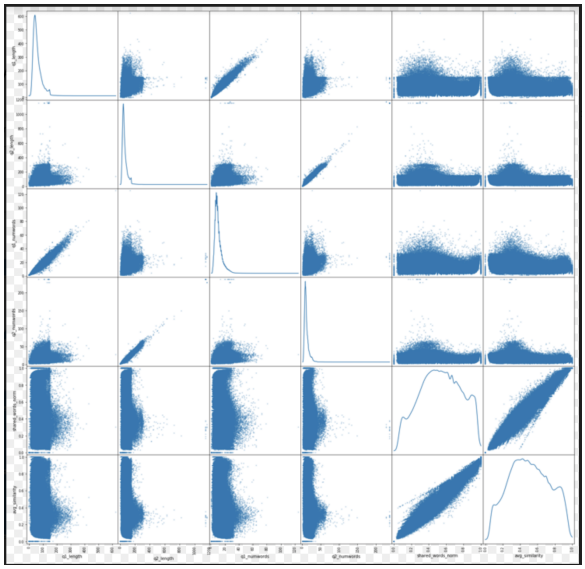

Figure 3: Visualizing features using Scatter matrix. Diagonal shows trend of individual features and other elements shows combined scatter plot of hand crafted features.

5.2 PREPROCESSING

After understanding the dataset we applied the following text preprocessing/cleaning techniques on the dataset. The preprocessed texts were passed to then the models.

- Text Expansion: This include expanding quotation marks. For example, don't to do not
- Punctuation Removal: Removed all punctuation characters except question mark
- Punctuation padding: Padding each punctuation character with a whitespace after each punctuation mark(except the question mark character)
- Stemming: We also reduce inflected(or sometimes derived) words to their word stem.

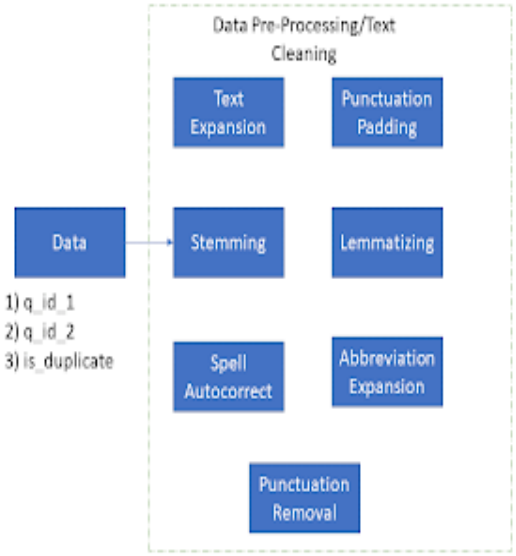

Figure 4: Preprocessing techniques applied.

- Lemmatizing: The process of reducing the word in context of Inflectional morphology from cats->cat.
- Spell autocorrect: Our preprocessor scans each word and autocorrect the spelling if required.
- Abbreviation Expansion: For example, converting U.S.A to United States of America.

5.3 Classifiers Using Handcrafted features

We trained four traditional Machine Learning models in order to establish a baseline before building more advanced models.
The following were the approaches we tried:

- TF-IDF Question Pair Similarity Scores + SVM
- TF-IDF Question Pair Similarity Scores + XgBoost
- TF-IDF Character Bigram/Trigram Count +XgBoost
- LDA Topic Modelling with its similarity +XgBoost

5.3.1 TF-IDF Question Pair Similarity Scores + SVM

In this model we used the tf-idf scores of the question pairs and used a linear SVM as a classifier to classify them as duplicate or not.

Vocabulary: Each question pair was extracted from the train data and a list of unique questions was generated. Each unique question was tokenized and tf-idf scores were computed considering this whole vocabulary setting.

Features: For the final SVM Classifier each word in the vocabulary was considered as a feature and the tf-idf scores of these words represented the feature values.

5.3.2 TF-IDF Question Pair Similarity Scores + XgBoost

The vocabulary and words(features) representation of the question is similar to our previous approach but instead of using SVM as a classifier we used XgBoost which is an adaptive gradient boosting algorithm based on random forests and decision trees.

5.3.3 TF-IDF Character Bigram/Trigram Count +XgBoost

In this model instead of operating on the word feature space we considered a character feature space. The intuition behind using a bigram and trigram sequence of characters was to extract a notion of semantic meaning out of these questions.

Vocabulary: Each question pair was extracted from the train data and a list of unique questions was generated. Each unique question was tokenized with character bigrams/trigrams and tf-idf scores were computed considering this whole vocabulary setting.

Features: For the final XGBoost Classifier each character bigram and trigram sequence in the vocabulary was considered as a feature in addition to the tf-idf scores.

5.3.4 LDA Topic Modelling with its similarity +XgBoost

In this model we used LDA (Latent Dirichlet Allocation) to group each unique questions in the training corpus into k topics (we empirically determined this k). The vocabulary was constructed by tokenizing on words for the unique questions. Each question is then represented as a probability distribution vector over the k extracted topics. Each of these topics are translated as features using their probability values and then a XGBoost classifier is used to classify them(like previous TF-IDF char based approach).

5.4 Deep Learning approaches

In recent years, Siamese recurrent architectures have become the obvious choice to identify semantic similarities in sentences[8]. LSTMs are naturally suited for text classification tasks of variable-length inputs.

Hence, we used the Siamese architecture in our experiments. Our networks have two identical sub-networks, one for each question. First, the question pairs are preprocessed and fed to an embedding layer. This is further propagated through a LSTM layer. The final vector representation of the sentences are compared using the Manhattan similarity function.

5.4.1 Word Embedding + LSTM

In this model, the two sub-networks in the Siamese architecture have an embedding layer followed by an LSTM layer. Specifically, we use Google's word2vec(300-dimension) embeddings in the initial embedding layer, and the LSTM layer has 50 hidden states. The weights between the two sub-networks are shared in both these layers. For training the network, we used the AdaDelta optimizer and the Mean Squared Error as the loss function.

5.4.2 Character Embedding + LSTM

This has the same network architecture as described above, however, we made use of GloVe Character Embeddings to get a representation of the questions in the embedding layer. The optimizer, loss function, and other hyperparameters are identical to the above network.

5.4.3 Word Embedding + BiLSTM

BiLSTM architectures have proved to extract the contextual information better than unidirectional LSTMs. Unidirectional LSTMs only preserves information for the past, but BiLSTM are capable of processing words appearing later in the questions.

5.4.4 Word Embedding + ConvID + Manhattan Distance

RNNs operate sequentially, the output for the second input depends on the first one and so we can't parallelise an RNN. Convolutions have no such problem, each "patch" a convolutional kernel operates on is independent of the other meaning that we can go over the entire input layer concurrently. Due to the above reason we have tried to incorporate conv1d to solve this problem. We have used normal word-embedding pretrained on google-news-dataset and two layers of

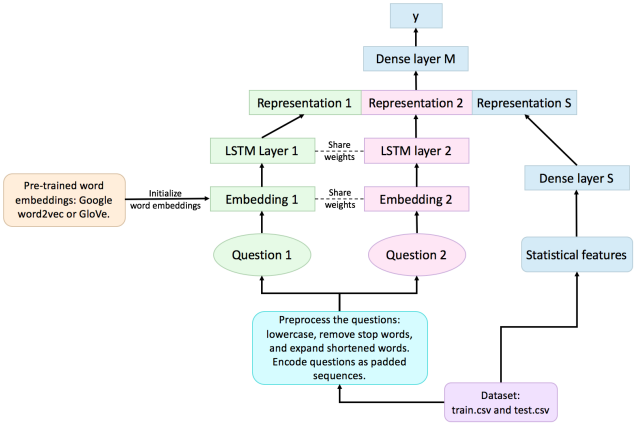

Figure 5: Word Embedding+TF-IDF Features + BiLSTM + Manhatten Distance

conv1d with kernel size 4. After that we added a layer to compute manhattan distance.

5.4.5 *Word Embedding + Handcrafted TF-IDF Features + BiLSTM + Manhatten Distance*

The bi-directional LSTM is good at understanding the sentence context as whole. But the performance was not as high as expected. To improve the accuracy, we merged the hand-crafterd TF-IDF features just before the last convolution layer. We tried adding topic modeling features also, but this was not giving much improvement. Hence, we didn't explore adding topic modeling further.

5.4.6 *BERT*

BERT has achieved state-of-the-art results in various tasks, including Question Answering, GLUE benchmark, and others[3]. We used a pre-trained BERT model (uncased_L-24_H-1024_A-16) for our classification task. A merge operation is performed before concatenating the features representation of the two models. This is propagated to a 2-neuron output layer where a sigmoid activation is used.

The pretrained model was further fine-tuned on the Quora Question Pairs dataset for 2 epochs. For the fine tuning, we use mean log loss and the Adam optimizer.

In this model, the words in the sentences are first tokenized. BERT uses word-piece tokenization for converting text to tokens. Tokenizer also takes care of text normalization, lowercases the input and strips out accent markers. The tokens are then converted to features by feeding into the pretrained model. These features are better at representing the context as BERT is deeply bidirectional and has been trained on an extremely large text corpus. Finally, these features are brought down to the output layer for the purpose of classification.

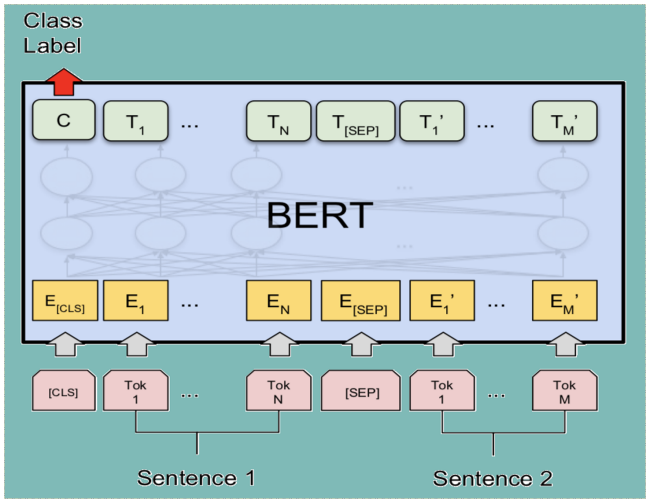

Figure 6: BERT Model

Model	Accuracy
TF-IDF+SVM	67.55
TF-IDF+XgBoost	70
Topic Modeling+XgBoost	72
Char Bigram/Trigram+XgBoost	80.05

Table 1: Classical Models

6 RESULT

6.1 Classical Models

With the unique questions as a training corpus, the TF-IDF model with its scores as weights and word tokens as features gave around 67.55 percent as a classification accuracy.Combining it with adaptive boosting using XgBoost as a classifier we pushed it to about 70%. Using a slightly different approach with topic modelling and topic vectors using LDA we achieved around 72 percent accuracy.

The main surprising result which we achieved with handcrafted features was with the TF-IDF model with character bigram and trigram features. Our intuition was that bigram and trigram would help us extract character sequences. Another key difference was that this approach used character sequences as features instead of word sequences. The classification accuracy we achieved was 80.05

Models	Accuracy(%)	F-1 Score(0,1)	Prescision(0,1)	Recall(0,1)	Log-Loss
BERT+Dense Layer	91.6	0.91,0.80	0.87,0.86	0.95,0.75	0.263
W2Vec+TF-IDF+BiLSTM+Manhattan	86.22	0.88,0.76	0.85,0.83	0.93,0.70	0.2739
TF-IDF Character Bigram+XgBoost	80.08	0.87,0.74	0.83,0.81	0.91,0.67	N/A

Table 2: Best Models

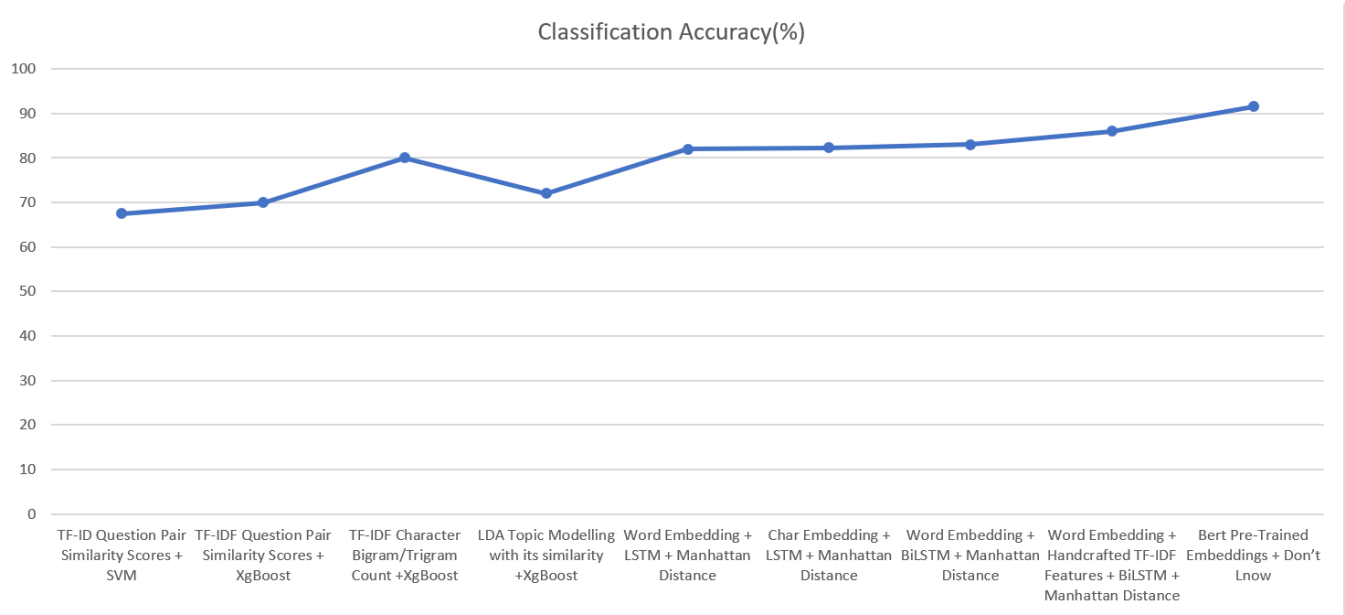

Figure 7: Result Accuracy Diagram

Key Conclusions and takeaways with classical approaches:

- Character representations work surprisingly well and in this case the bigram/trigram sequences outperformed the word sequences.
- These models are fairly simple and easy to deploy as we did in our demo application.
- So, when we tried neural approaches we can try both character and word embedding representations of them.
- Through bigram and trigram sequences the consecutive appearance was easily captured but worlds like rich and wealthy were not treated as same.

6.2 Neural Models

As seen in the table, all deep learning approaches outperform the classical Machine Learning techniques. We attribute this to the semantic relationships that these models are able to understand. In other words, the traditional models are trained using superficial features which don't capture the underlying meaning of the questions.

Model	Accuracy
W2Vec+LSTM+Manhattan	82
W2Vec+ConvID+Manhattan	82.75
Char2Vec+LSTM+Manhattan	82.32
W2Vec+BiLSTM+Manhattan	83
W2Vec+TF-IDF+BiLSTM+Manhattan	86.22
W2Vec+Bert	91.6

Table 3: Neural Models

Furthermore, we see that the classification accuracy further increases in BiLSTM architectures. This further verifies that these architectures are learning the contextual information, which is key for classifying duplicate questions.

Finally, we could see that the BERT model achieves a classification accuracy of 91.6%. This simply implies

that the features produced by BERT represents the questions better when compared to the other embeddings we tried. Since the model is deeply bidirectional and has been trained on a huge data corpus, it is able to identify latent relationships effectively.

6.3 Examples-Demo Application

In an attempt to build a practical systems we deployed the simple character bigram/trigram model, the ConvID + LSTM model and the Character Embedding Model with LSTM in an application. Some of the examples question we tried include: ['how old are you?', 'how can a person reduce weight?', 'does democracy benefit humanity?', 'how to become rich?']. When we gave a similar question like 'how to become wealthy', all the models gave a high similarity score for the question : 'how to become rich?'. The models were able to successfully link rich to wealthy.

7 CONCLUSION

The deep learning models outperformed the classical approaches in understanding the semantic relationships between the questions. By using BERT the pre-trained model(fine-tuned for our use case) with a simple feed forward neural network we achieved 91.6 % classification accuracy on the validation set.

Reasons why we feel that BERT outperformed all other models:

- The model has been trained on good quality data corpus and it understands how questions are phrased and written. BERT is more context aware than other embedding representations.
- The bidirectional nature of BERT is able to efficiently extract the semantic meaning out of similar questions.

8 CHALLENGES

- One of our major challenges was deploying BERT.Due to the large model complexity, the inference and prediction of questions took a lot of time
- Due to the large size of the dataset the training time was always on the higher side for all our models.
- Data pre-processing was a challenge due to the large use of abbreviations and text shortenings in the training questions.

9 FUTURE WORK

- We can augment our data by flipping the question pairs to supply our models with more training data
- Many question pairs where labelled wrongly. If we manually re-label these set of questions, the results might improve further
- We can add a more deeper network after the BERT representations are extracted instead of a simple feed forward network which we have currently implemented.

REFERENCES

[1] Elkhan Dadashov, Sukolsak Sakshuwong, and Katherine Yu. 2017. Quora Question Duplication.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[4] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *HLT-NAACL*.

[5] Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. [n. d.]. UMBC EBIQUITY-CORE: Semantic Textual Similarity Systems. ([n. d.]).

[6] Shuo Chang Lili Jiang and Nikhil Dandekar. 2017. Semantic Question Matching with Deep Learning. (2017).

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases/-and-their-compositionality.pdf

[8] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. (2016), 2786–2792. http://dl.acm.org/citation.cfm?id=3016100.3016291

[9] M. Schuster and K.K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.* 45, 11 (Nov. 1997), 2673–2681. https://doi.org/10.1109/78.650093

[10] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *CoRR* abs/1503.00075 (2015). arXiv:1503.00075 http://arxiv.org/abs/1503.00075