Hungarian Layer: Logics Empowered Neural Architecture

Han Xiao

Abstract

Neural architecture is a purely numeric framework, which fits the data as a continuous function. However, lacking of logic flow (e.g. if, for, while), traditional algorithms (e.g. Hungarian algorithm, A* searching, decision tree algorithm) could not be embedded into this paradigm, which limits the theories and applications. In this paper, we reform the calculus graph as a dynamic process, which is guided by logic flow. Within our novel methodology, traditional algorithms could empower numerical neural network. Specifically, regarding the subject of sentence matching, we reformulate this issue as the form of task-assignment, which is solved by Hungarian algorithm. First, our model applies BiLSTM to parse the sentences. Then Hungarian layer aligns the matching positions. Last, we transform the matching results for softmax regression by another BiLSTM. Extensive experiments show that our model outperforms other state-of-the-art baselines substantially.

1. Introduction

Neural architecture, inspired by brain science, has a long history (McCulloch & Pitts, 1943). This branch of machine learning grows from single perception (Casper et al., 1969) to deep complex network (Lecun et al., 2015), achieving many great successes (Silver et al., 2016). Mathematically, all the operations in neural architecture (i.e. matrix multiply, non-linear function, convolution, etc.) are numerical and continuous. which could benefit from back-propagation algorithm (Rumelhart et al., 1988).

However, logic flow (i.e. “if, for, while” in the sense of programming language) is the foundation of computations, without which intelligence system is incomplete in theory and imperfect in performance. Traditional option treats logic flow as a branching and non-differentiable structure, which is incompatible with back-propagation and optimization framework.

In this paper, our research solves the challenges by treating logic flow as a dynamically graph-constructing process, where forward propagation would dynamically construct the graph, then backward propagation perform as usual in this constructed graph without any non-differentiable issue.

Referring to Figure 1, we focus on an illustrative example of question answering, where each sentence could be a knowledge query or common chat. Thus, we plug a logics empowered layer between the encoder and decoder. Specifically, if the representation of sentence is near the knowledge prototype vector, we transform this representation by knowledge-specific matrix, otherwise we perform the chatting-specific transformation. Finally, the decoder generates the answer according to the transformed representation.

In the forward propagation, our methodology activates some branch dynamically according to the condition, then dynamically construct the graph according to the instructions in the active branch. As to Figure 1, assuming the “else-branch” is activated, $\text{sigmoid}(Mq)$ rather than $\text{sigmoid}(Wq)$ is dynamically connected to $\text{rep}$. No matter which branch is turned on, $\text{rep}$ is parsed by LSTM to generate the result. In this way, the forward pass has dynamically constructed a continuous and differentiable graph without branching, which could be processed by backward propagation and optimization framework. Specifically, the backward pass propagates from decoder to chatting linear layer, then to encoder, where knowledge linear layer is deactivated from the constructed graph for this sample.

It is worthy to note that “for, while” could be reformed as “if, goto”. “goto” means to perform “if” in multiple times, which could also be processed by above ideas. In essence, logics is a functionality of selection, which guides the forward propagation and harms no gradient calculation of neural parts in backward propagation.

In fact, all of the traditional algorithms could be formulated as the combination and iteration of “if, while, for”. Mathematically, on the other word, any algorithms are organized by repeat (i.e. while, for) and branch (i.e. if). Therefore,
all of the traditional algorithms could be embedded into neural architectures with our proposed principle, such as Hungarian algorithm, max flow algorithm, A* searching, resolution method, decision tree algorithm, K-MEANS, etc.

Specifically, in this paper, we embed the Hungarian algorithm (Wright, 1990) into neural architecture as Hungarian layer (Algorithm 1) for the task of sentence matching. Hungarian algorithm tackles the task-assignment problem, which corresponds to the alignment in sentence matching, illustrated in Figure 2. Simply, Hungarian algorithm works out the alignment relationship, according to which, the sentence representations dynamically connect to the next layer.

Regarding sentence similarity, we conjecture only the aligned unmatched parts are semantically critical. For the example of the sentence pair: "NASDAQ raises two point in the last Friday" and "NASDAQ falls two point in the last Friday", the matched parts (i.e. Friday, last, two point, etc.) barely make contribution to the semantic sentence similarity, but the unmatched part (i.e. "raise" and "fall") determines these two sentences are semantically dissimilar.

Traditional alignment method takes advantage of attention mechanism (Wang et al., 2017), which is a soft-max weighting technique. Weighting technique could pick out the most similar part, but is weak in modeling sentence-specific dissimilar part. For above example, “point” has an extremely low weight with “NASDAQ”, but this is not the sentence-specific dissimilar evidence, because the comparison between these two words is inappropriate. However, Hungarian layer assigns the aligned pair with exclusiveness, the word “raise” would be assigned to “fall” with a negative similarity, making a strong evidence for discrimination.

We conduct our experiments on public benchmark datasets: "Quora Paraphrase Dataset" for paraphrase detection and WikiQA dataset for answer selection. Experimental results demonstrate that our model outperforms other baselines extensively and significantly, which illustrates the effectiveness of our methodology. The most important conclusion is that “our model is differentiable”, which verifies our theory and provides a novel methodology to neural architecture.

Contribution. (1.) We provide the principle for embedding traditional algorithms into neural architectures, which opens a novel methodology to the theories and applications of artificial intelligence. Also, this principal idea may trigger new generation programming languages. (2.) We offer a new perspective for sentence matching, which focuses on the aligned unmatched parts of two sentences. Accordingly, we propose the Hungarian layer to extract the aligned unmatched parts. (3.) Our model outperforms other baselines extensively, verifying the effectiveness of our theory and method.

Organization. In the Section 2, we survey the related work of sentence matching. In the Section 3, we introduce our methodology and our neural architecture. In the Section 4, we conduct the experiments for performance and verification. In Section 5, we list the potential future work from a developing perspective. In the Section 6, we conclude our paper and publish our codes.
2. Related Work

We have surveyed this task, and categorized related papers into three lines: non-neural architecture, neural architecture with independent sentence encoder and neural architecture with interdependent sentence encoder.

2.1. Non-Neural Architecture

The topic of sentence matching raises in the last decade, because many applications, such as paraphrase detection (Yin et al., 2015) and answer selection (Wang et al., 2017), could directly benefit from sentence pair comparison in natural language processing. The development has been through four stages before neural architectures: word specific, syntactic tree specific, semantic matching and probabilistic graph modeling.

Firstly, (Bilotti et al., 2007) focuses on simple surface-form matching between bag-of-words, which produces poor accuracy, because of word ambiguities and syntactic complexity. Therefore, syntactic analysis is introduced into this task for semantic understanding, such as deeper semantic analysis (Shen & Lapata, 2007), quasi-synchronous grammars (Wang et al., 2009) and tree edit distance (Heilman & Smith, 2010). Notably, most of these methods compare the grammar tree (e.g. syntactic tree, dependency tree, etc.) of sentence pair. Further, semantic information such as negation, hypernym, synonym and antonym relations is integrated into this task for a better prediction precision, (Lai & Hockenmaier, 2014). Finally, (Yao et al., 2013) leverages a semi-Markov CRF to align phrases rather than words, which consumes too many resources for industrial application.

In summary, the advantage of this branch is to model alignment or correlation in the encoding process, to some extent.

2.2. Neural Architecture: Independent Sentence Encoder

With the popularity of deep neural network, some neural architectures are proposed to analyze the complex language phenomenon in a data-fitting way, which promotes the performance. First of all, the neural network extracts the abstracted feature from each sentence independently, then measures the similarity of abstracted feature pair. There list three frameworks: CNN-based, RAE-based and LSTM-based.

Commonly, CNN could be treated as n-gram method, which corresponds to language model. Specifically, (Yu et al., 2014) applies a bi-gram CNN to jointly model question and answer candidates. (Yang et al., 2015) achieves the state-of-the-art performance by following this work. (Yih et al., 2013) is also based on CNN, but to tackle paraphrase identification problem. Also for paraphrase detection, (Socher et al., 2011) has proposed a RAE based model to characterize phrase-level representation, which promotes simple embedding pooling method, (Blacoe & Lapata, 2012). Besides, for entailment inference, (Marelli et al., 2014) employs the recursive neural network to encode sentence semantics, while (Rockt?schel et al., 2015) takes the idea of attention mechanism to strengthen LSTM for this task. Note that, (Rockt?schel et al., 2015) is not a purely independent feature encoder, because this model measures the correlations in encoding process, to some extent.

In summary, the advantage of this branch is to model complex and ambiguous linguistic phenomenon in a black-box style. However, the disadvantage is that the encoder could not adjust the abstracted representations according to the correlation of sentence pair, making an imperfect matching process.

2.3. Neural Architecture: Interdependent Sentence Encoder

To emphasize the correlation of sentence pair in sentence encoder, the researchers propose attention-based neural architecture, which guides the encoding process according to the corresponding part. There introduce two representative work: ABCNN (Yin et al., 2015) and BiMPM (Wang et al., 2017).

ABCNN is a CNN-based model. In a single stage, this model computes the attention similarity matrix for the convolution layer, then sums out each row and column as the weighs of pooling layer. The output of convolution layer is weighted by pooling layer in an average manner as the output of this stage. ABCNN could stack at most three stages. This method achieves satisfactory performance in many tasks, because of modeling correlation in sentence encoder.

BiMPM is a LSTM-based model, which involves many tricks (e.g. dropout, character embedding, etc.). Basically, the word embeddings of each sentence are parsed by a parameter-shared bi-directional LSTM (BiLSTM). Then the outputs are passed by the matching layer, which includes four alignment methods (i.e. full matching, max-pooling matching, attentive matching, max-attentive matching). Last, the matching scores from matching layer, are parsed by another bi-directional LSTM (BiLSTM) to make the sentence representation for the soft-max regression. This method holds the state-of-the-art performance, because of integrating four alignment methods and applying complex network structures.

In summary, the advantage of this branch is to model alignment or correlation in the encoding process. However, the
Hungarian Layer: Logics Empowered Neural Architecture

Figure 3. Our Neural Architecture. The sentence pair is \((p_1, p_2, ..., p_M)\) and \((q_1, q_2, ..., q_N)\). First, word embeddings are parsed by bi-directional LSTM. Then, the hidden representations are processed by Hungarian layer, the outputs of which correspond to the concatenation of aligned representations weighted with cosine dissimilarity \(^1\). Last, the results of Hungarian layer are transformed by another bi-directional LSTM for soft-max regression.

Disadvantage is to focus on the matched part, rather than the unmatched part that is critical in this task as previously discussed.

3. Methodology

Firstly, we introduce the basic components of our neural architecture. Secondly, we discuss the objective. Finally, we analyze the implementation, that how to dynamically construct this specific graph.

3.1. Neural Architecture

This neural architecture of our model is illustrated in Figure 3, which is composed by five components, namely, word embedding, sentence-specific bi-directional LSTM, Hungarian layer, alignment-specific bi-directional LSTM and soft-max regression.

Word Embedding. The goal of this layer is to represent each word \(w_{s,i}\) in every sentence \(s\) with \(d\)-dimensional semantic vectors. The word representations, which are pretrained by GloVe (Pennington et al., 2014), are unmodified within the learning procedure. The inputs of this layer are a pair of sentences as word sequences \(p = (p_1, p_2, ..., p_M)\) and \(q = (q_1, q_2, ..., q_N)\), while the outputs are corresponding embedding matrices as \(P = (p_1, p_2, ..., p_M)\) and \(Q = (q_1, q_2, ..., q_N)\).

Sentence-Specific Bi-Directional LSTM. The purpose of this layer is to transform word representations to contextual representations. For contextual encoding, we employ a parameter-shared bi-directional LSTM (Hochreiter & Schmidhuber, 1997) to parse the word embeddings into hidden representations, mathematically as:

\[
\begin{align*}
    h^f_i &= \text{LSTM}^{\text{forward}}(h_{i-1}, s_i) \\
    h^b_i &= \text{LSTM}^{\text{backward}}(h_{i+1}, s_i)
\end{align*}
\]

where \(h_i = [h^f_i, h^b_i]\) are the hidden representations and \(s_i\) corresponds to the \(i\)-th word in the source/target sentence or \(p_i,q_i\).

Hungarian Layer. This layer, that is the logic component of our model, extracts the aligned parts from source and target sentences. This layer is composed by two sequential stages. Algorithm 1 demonstrates the first stage. First, the hidden representations are crossed to generate the similarity matrix \(M_{i,j}\). Then, Hungarian algorithm works out the alignment information with this similarity matrix. Last, hidden representations and corresponding similarities are dynamically connected to the second stage, according to the alignment information.

The second stage attempts to extract the aligned unmatched hidden representations, which means, if two aligned repre-
sentations are too similar, the weights for them should be small, otherwise, large dissimilarity leads to large weights. For this reason, we introduce cosine dissimilarity, mathematically as:

\[ \alpha_i = 1 - m_i \]  

where \( \alpha_i \) is the weight of \( i \)-th aligned representation and \( m_i \) is the matched similarity. Thus, the unmatched hidden representations are concatenated hidden representations, which are weighted by cosine dissimilarity:

\[ a_i = \alpha_i \otimes [h^p_i, h^q_i] \]  

where \( a_i \) is the output of Hungarian layer, \( h^p_i/h^q_i \) are the source/target hidden representations and \( \otimes \) is the element-wise multiplication.

Notably, \( a_i \) corresponds to the ascend order of \( m_i \), mathematically, \( m_1 < m_2 < ... < m_N \). The first aligned position is most possible unmatched, while the last one is most possible matched. Also note that we pad the source/target sentence into the same length.

Algorithm 1 Hungarian Layer: First Stage

Input: Target and source sentence representations of \( \{s_i\} \) and \( \{t_i\} \).
Output: \( \{(a_i, b_i, m_i)\} \), where \( a_i, b_i \) means the matched hidden representation pair for source and target respectively, and \( m_i \) means matched similarity.

1: Generate the similarity matrix:
\[ m_{i,j} = \frac{\text{dot}(s_i, t_j)}{|s_i||t_j|} \]

2: for all \( i \in [1...|S|] \), where \( S \) is the maximum length of source/target sentence. do
3: Compute \( m_i \), whose position corresponds to \( (p_i, q_i) \) and it is the maximum of similarity matrix.
4: Delete \( p_i \) row and \( q_i \) column from the similarity matrix.
5: end for
6: return \( \{(a_i = s_{p_i}, b_i = t_{q_i}, m_i)\} \).

Alignment-Specific Bi-Directional LSTM. This layer transforms the aligned unmatched representations into final discriminative features. Also, the bi-directional LSTM is employed in this layer as described before. The input of this layer is the output of Hungarian layer, while the output of that is hidden feature representations as \( h_i^{\text{final}} \).

Soft-Max Regression. This layer averages the output of alignment-specific BiLSTM as the final discriminative feature. A soft-max layer transforms the feature into probabilistic distributions of target, such as

\[ f = \frac{1}{|S|} \sum_{i=1}^{|S|} h_i^{\text{final}} \]

\[ P(y|f) = \frac{\exp(w_y^T f)}{\sum_{i=1}^{|Y|} \exp(w_y^T f)} \]

where \( y \) is the target of our model, \( |S| \) is the length of source/target sentence and \( w_y \) is the target-specific weight of soft-max distribution.

3.2. Objective

For the reason of numerical stability, we choose the absolute loss as our objective, as:

\[ L = \sum_{i=1}^{|D|} |y_i - l_i| \]

where \( y \) is the target output of our model, \( |D| \) is the total number of training set and \( l \) is the data label.

3.3. Dynamically Constructed Graph

Figure 4. Illustration of Dynamically Constructed Graph. In the sub-figure (a), the graph is static, where naturally sequential alignment is applied such as the \( i \)-th word in target corresponding to \( i \)-th word in source. In the sub-figure (b), Hungarian layer computes the alignment information, according to which, the hidden representations are dynamically connected to the next layer. For example, \( (1 \rightarrow 2) \) means 1st word in target corresponds to 2nd word in source.

Previously discussed, the logic components are embedded into neural architecture by dynamically constructing the
Hungarian Layer: Logics Empowered Neural Architecture

calculus graph. In this subsection, we illustratively exemplify how the graph is dynamically constructed in Hungarian layer as Figure 4 shows.

Once the graph is dynamically constructed in forward pass, the backward process could propagate from the links between layers, without any branching and non-differentiated issues. In this way, the optimization framework could still adjust the parameters of neural architectures.

4. Experiment

In the section, we verify our model performance on two tasks: paraphrase identification and entailment inference, which are directly applications of sentence matching. We first introduce the experimental settings, in Section 4.1. Then, in Section 4.2 and 4.3, for performance evaluation, two benchmark tasks \( (i.e. \text{paraphrase identification}, \text{entailment inference}) \) would testify our model. Last, in order to further testify our assumptions, that the aligned unmatched parts are semantically critical, we conduct a case study for illustration.

4.1. Experimental Setting

We initialize the word embedding with 300-dimensional GloVe (Pennington et al., 2014) word vectors pre-trained in the 840B Common Crawl corpus (Pennington et al., 2014). For the out-of-vocabulary (OOV) words, we directly apply zero vector as word representation. We set the hidden dimension as 100 for each BiLSTM. To train the model, we leverage AdaDelta (Zeiler, 2012) as our optimizer, with hyper-parameter as moment factor \( \eta = 0.6 \) and \( \epsilon = 1 \times 10^{-6} \). We train the model until convergence, but at most 100 rounds. Regarding the batch size, we always choose the biggest one to fully utilize the computing devices.

4.2. Paraphrase Identification

Motivation. Given two similar sentences, the model should discriminate whether they tell the same story, which is the motivation of this task, \( ( \text{Yin et al., 2015}) \). This task could directly benefit natural language processing, such as detecting redundancy Quora questions.

Dataset. Actually, to demonstrate the effectiveness of our model, we just perform our experiments on the dataset of “Quora Question Pairs”\(^1\). To be a fair comparison, we follow the splitting rules of (Wang et al., 2017). Specifically, There are over 400,000 question pairs in this dataset, and each question pair is annotated with a binary value indicating whether the two questions are paraphrase of each other or not. The authors randomly select 5,000 paraphrases and 5,000 non-paraphrases as the development set, and sample another 5,000 paraphrases and 5,000 non-paraphrases as the test set. They keep the remaining instances as the training set. Regarding the details of dataset, please refer to (Wang et al., 2017) \(^2\).

Baselines. To make a sufficient comparison, we choose six state-of-the-art baselines: Siamese CNN, Multi-Perspective CNN, Siamese LSTM, Multi-Perspective LSTM, L.D.C. and BiMMPM. Specifically, Siamese CNN and LSTM encode the input two sentences into two sentence vectors by CNN and LSTM, respectively, \( ( \text{Wang et al., 2016a}) \). Based on the two sentence vectors, a cosine similarity is leveraged to make the final decision.

(Wang et al., 2017) proposes the multi-perspective technique, which computes the similarity in multiple aspects. Within this technique, Siamese architectures are promoted as Multi-Perspective architectures. L.D.C model \( ( \text{Wang et al., 2016b}) \) is an attention-based method, which decomposes the hidden representations into similar and dissimilar parts. L.D.C is still a powerful model which achieves many state-of-the-art performance. Regarding BiMMPM, please refer to Section 2.

Table 1. Performance Evaluation on Quora Question Dataset.

| Methods                  | Accuracy (%) |
|--------------------------|--------------|
| Siamese CNN \( (\text{Wang et al., 2016a}) \) | 79.60        |
| Multi-Perspective CNN    | 81.38        |
| Siamese LSTM \( (\text{Wang et al., 2016a}) \) | 82.58        |
| Multi-Perspective LSTM   | 83.21        |
| L.D.C. \( (\text{Wang et al., 2016b}) \)        | 85.55        |
| BiMMPM \( (\text{Wang et al., 2017}) \)          | 85.88        |

| Our Model                | 86.11        |

Results. Our results are reported in Table 1. We could conclude that:

1. Our method outperforms all the baselines, achieving the state-of-the-art performance. This comparison illustrates the effectiveness of our model.

2. Compared with BiMMPM \( (\text{Wang et al., 2017}) \), which ensembles four alignment methods, Hungarian layer.

\(^1\) The url of the dataset: https://data.quora.com

\(^2\) Baseline codes and datasets: https://zhiguowang.github.io/
is much better at alignment, because our model apply exclusive matching strategy. The exclusiveness stems from logics empowered neural architecture.

3. Compared with L.D.C. (Wang et al., 2016b), which still analyzes the dissimilar part, our model could exactly extract the aligned unmatched parts rather than fuzzy dissimilar part, which promotes the performance.

4.3. Answer Selection

**Motivation.** Given a question, the model should pick out the correct answers from several candidate ones, which is the motivation of this task. Specifically, The task is to rank the candidate answers based on their relatedness to the question. Evaluation measures are mean average precision (MAP) and mean reciprocal rank (MRR). This task is a simplification of question answering, which is foundational research in natural language understanding.

**Dataset.** WikiQA is the public benchmark dataset with open domain queries. Following the related work (Yin et al., 2015), we assume that there is at least one correct answer for one question. Specifically, this dataset consists of 20,360 question-answer pairs in training set, 1,130 pairs in development set and 2,352 pairs in test set. As usual, we ignore the questions without correct answers in test set.

**Baselines.** To make a sufficient comparison, we choose several baselines listed in Table 2. (Yang et al., 2015) is non-neural architecture, which counts the occurring count of non-stop words between question and answer with weights. Also, ABCNN (Yin et al., 2015) and BiMPM (Yang et al., 2015) are two state-of-the-art baselines for this task, introduced in Related Work (Yin et al., 2015).

**Implementation.** For a fair comparison with relevant methods, we directly report the results under the same setting from the literature. However, for a fair setting, we still implement the BiMPM method without any extra tricks (e.g. character embedding, dropout, etc.). Regarding our model, we barely use tricks, which could be easily repeated for further researches.

**Results.** Our results are reported in Table 2. There are several observations listed as:

- Our method beats all the baselines, achieving the state-of-the-art performance. Experimental results demonstrate the effectiveness of our model.
- Compared with Word Count (Yang et al., 2015), which is a non-neural architecture, our model takes neural structures to extract more semantics from the data, leading to better performance.
- Compared with ABCNN, which is attentive aligned model based on CNN, Hungarian layer is much better

| Methods                  | MAP   | MRR   |
|--------------------------|-------|-------|
| Word Count (Yang et al., 2015) | 0.652 | 0.665 |
| ABCNN (Yin et al., 2015)   | 0.692 | 0.711 |
| (Miao et al., 2015)       | 0.689 | 0.707 |
| (Wang & Ittycheriah, 2015) | 0.706 | 0.723 |
| (He & Lin, 2016)          | 0.709 | 0.723 |
| (Rao et al., 2016)        | 0.701 | 0.718 |
| BiMPM (Wang et al., 2017) | 0.718 | 0.731 |
| Our Model                | 0.724 | 0.738 |

at alignment, because our model exclusively extracts the aligned matched and unmatched parts. All of the critical functionalities are based on logics, which are embedded within our proposed principle.

5. Future Work

For future work, there are three lines: design new logic layer, implement a dynamic calculus graph framework and develop a new generation programming language.

Previously discussed, all the traditional algorithms could be embedded into neural architectures, which means, the algorithms such as max flow algorithm, A∗ searching, resolution method, PSP, decision tree, gradient boosting, FP-growth could be embedded into neural systems. On the other hand, neural structures could strengthen these algorithms in a strong data-fitting manner.

Actually, current calculus graph framework such as TensorFlow (Abadi et al., 2016) could not implement logics empowered structures. Also, only the GPU could not complete this task. We need a CPU-GPU jointly enhanced dynamic calculus graph framework to implement logics empowered neural architectures.

Finally, logics empowered neural architecture is a learnable Turing machine, which opens a new branch for the theory of computation. Based on the novel paradigm, we may design a new generation programming language, which is further influential.

6. Conclusion

This paper proposes the principle to embed logics-based algorithms into neural architectures. Based on this principle, we leverage Hungarian algorithm to design Hungarian layer, which extracts the aligned matched and unmatched parts exclusively from sentences pair. Then a model is designed by assuming the unmatched parts are semantically critical. Experimental results on benchmark datasets demonstrate the effectiveness of our proposed methods.
References

Abadi, Martn, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, and Devin, Matthieu. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016.

Bilotti, Matthew W., Ogilvie, Paul, Callan, Jamie, and Nyberg, Eric. Structured retrieval for question answering. In International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 351–358, 2007.

Blacoe, William and Lapata, Mirella. A comparison of vector-based representations for semantic composition. In Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 546–556, 2012.

Casper, M, Mengel, M, Fuhrmann, C, Herrmann, E, Appenrodt, B, Schiedermaier, P, Reichert, M, Bruns, T, Engelmann, C, and Grnhage, F. Perceptrons: An introduction to computational geometry. 75(3):3356–62, 1969.

He, Hua and Lin, Jimmy. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 937–948, 2016.

Heilman, Michael and Smith, Noah A. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA, pp. 1011–1019, 2010.

Hochreiter, Sepp and Schmidhuber, Jrgen. Long short-term memory. Neural Computation, 9(8):1735, 1997.

Lai, Alice and Hockenmaier, Julia. Illinois-lh: A denotational and distributional approach to semantics. In International Workshop on Semantic Evaluation, pp. 329–334, 2014.

Lecun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. Nature, 521(7553):436–444, 2015.

Marelli, Marco, Menini, Stefano, Baroni, Marco, Bentivogli, Luisa, Bernardi, Raffaela, and Zamparelli, Roberto. A sick cure for the evaluation of compositional distributional semantic models. In Language Resources and Evaluation Conference, pp. A–696, 2014.

McCulloch, Warren S and Pitts, Walter H. A logical calculus of ideas imminent in nervous activity. 1943.

Miao, Yishu, Yu, Lei, and Blunsom, Phil. Neural variational inference for text processing. Computer Science, pp. 1791–1799, 2015.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. Glove: Global vectors for word representation. In Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543, 2014.

Rao, Jinfeng, He, Hua, and Lin, Jimmy. Noise-contrastive estimation for answer selection with deep neural networks. In ACM International on Conference on Information and Knowledge Management, pp. 1913–1916, 2016.

Rockt'schel, Tim, Grefenstette, Edward, Hermann, Karl Moritz, Ko'sisky, Tom?, and Blunsom, Phil. Reasoning about entailment with neural attention. 2015.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. MIT Press, 1988.

Shen, Dan and Lapata, Mirella. Using semantic roles to improve question answering. In EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic, pp. 12–21, 2007.

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, Driessche, George Van Den, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, and Lanctot, Marc. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484, 2016.

Socher, Richard, Huang, Eric H, Pennin, Jeffrey, Manning, Christopher D, and Ng, Andrew Y. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In Advances in Neural Information Processing Systems, pp. 801–809, 2011.

Wang, Mengqiu, Smith, Noah A., and Mitamura, Teruko. What is the jeopardy model? a quasi-synchronous grammar for qa. In EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic, pp. 22–32, 2009.

Wang, Zhiguo and Ittycheriah, Abraham. Faq-based question answering via word alignment. 60(3):1810–1815, 2015.

Wang, Zhiguo, Mi, Haitao, and Ittycheriah, Abraham. Semi-supervised clustering for short text via deep representation learning. 2016a.
Wang, Zhiguo, Mi, Haitao, and Ittycheriah, Abraham. Sentence similarity learning by lexical decomposition and composition. 2016b.

Wang, Zhiguo, Hamza, Wael, and Florian, Radu. Bilateral multi-perspective matching for natural language sentences. 2017.

Wright, M. B. Speeding up the hungarian algorithm. *Computers and Operations Research*, 17(1):95–96, 1990.

Yang, Yi, Yih, Wen-tau, and Meek, Christopher. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pp. 2013–2018, 2015.

Yao, X., Durme, B. Van, Callison-Burch, C., and Clark, P. Semi-markov phrase-based monolingual alignment. 2013.

Yih, Wen Tau, Chang, Ming Wei, Meek, Christopher, and Pastusiak, Andrzej. Question answering using enhanced lexical semantic models. In *Meeting of the Association for Computational Linguistics*, pp. 1744–1753, 2013.

Yin, Wenpeng, Schtze, Hinrich, Xiang, Bing, and Zhou, Bowen. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Computer Science*, 2015.

Yu, Lei, Hermann, Karl Moritz, Blunsom, Phil, and Pulman, Stephen. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

Zeiler, Matthew D. Adadelta: An adaptive learning rate method. *Computer Science*, 2012.