Extending Neural Question Answering with Linguistic Input Features

Fabian Hommel[1] **Matthias Orlikowski**[1] **Philipp Cimiano**[1,2] **Matthias Hartung**[1]
[1]Semalytix GmbH, Bielefeld, Germany
[2]Semantic Computing Group, Bielefeld University, Germany
`first.last@semalytix.com`

Abstract

Considerable progress in neural question answering has been made on competitive general domain datasets. In order to explore methods to aid the generalization potential of question answering models, we reimplement a state-of-the-art architecture, perform a parameter search on an open-domain dataset and evaluate a first approach for integrating linguistic input features such as part-of-speech tags, syntactic dependency relations and semantic roles. The results show that adding these input features has a greater impact on performance than any of the architectural parameters we explore. Our findings suggest that these layers of linguistic knowledge have the potential to substantially increase the generalization capacities of neural QA models, thus facilitating cross-domain model transfer or the development of domain-agnostic QA models.

1 Introduction

Recently, deep neural network approaches for question answering (QA) have gained traction. The strong interest in this task may be explained by two promises that resonate in neural QA approaches: For one thing, QA is claimed to bear the potential to subsume a lot of other NLP challenges. From this perspective, almost every task can be framed as a natural language question (Kumar et al., 2016). Thus, a QA model with the capacity to learn mappings from natural language terminology to formal linguistic concepts could be used as a *surrogate model*, reducing annotation and training effort and providing fast solutions to potentially complex NLP problems. For another, QA systems have always been considered as intuitive natural language interfaces for *information access* in various domains of (technical) knowledge.

As any other practical NLP solution targeting specialized domains, QA systems face the inherent challenges of cross-domain generalization or domain adaptation, respectively. However, QA approaches can be considered particularly suitable for this kind of problem, as the semantic underpinnings of question/answer pairs capture a universal layer of meaning that is domain-agnostic to some extent (but might require fine-tuning wrt. particular domain concepts or terminology).

We hypothesize that a promising approach towards rapid information access in specialized domains would be (i) to learn the aforementioned universal meaning layer from large collections of open-domain question/answer pairs, and (ii) adapt the resulting meaning representations to more specific domains subsequently. In this paper, we focus on the first problem.

Our work is based on the assumption that rich representations of linguistic knowledge at high levels of syntactic and semantic abstraction facilitates neural NLP models to capture "universal", domain-agnostic meaning, which in turn fosters performance in open-domain QA. Against this backdrop, we evaluate the impact of explicitly encoded linguistic information in terms of part-of-speech tags, syntactic dependencies and semantic roles on open-domain performance of a state-of-the-art neural QA model. We find that our re-implementation of the deep neural QANet architecture (Yu et al., 2018) benefits considerably from these linguistically enriched representations, which we consider a promising first step towards generalizable, rapidly adaptable QA models.

2 Related Work

In recent years, research about feature engineering for NLP models has subsided to some extent. This might be attributed to the ability of neural networks to perform hierarchical feature learning

(Bengio, 2009). Using neural approaches, many of the core NLP tasks like part-of-speech (PoS) tagging (Koo et al., 2008), dependency parsing (Chen and Manning, 2014), named entity recognition (Lample et al., 2016) and semantic role labelling (Roth and Woodsend, 2014; Zhou and Xu, 2015) have been improved. However, recent papers that make use of the improved performance in these areas are few (Alexandrescu and Kirchhoff, 2006; Sennrich and Haddow, 2016). Thus, we want to evaluate whether adding linguistic information to the inputs of a QA model improves the performance. Our approach to integrating linguistic input features by embedding each individually and concatenating the embeddings is inspired by Sennrich and Haddow (2016), who apply this approach in the context of machine translation.

This paper builds upon a host of recent developments in neural architectures for question answering or reading comprehension. While most approaches rely heavily on recurrent layers (Huang et al., 2017; Hu et al., 2018; Seo et al., 2016; Shen et al., 2017; Wang et al., 2017; Xiong et al., 2016), we chose to reimplement QANet, a self-attention based architecture (Yu et al., 2018).

Apart from that, we use the tools from Roth and Woodsend (2014) for extracting semantic roles over the whole Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016).

3 Extending QANet with Linguistic Input Features

As a testbed in order to assess the impact of linguistic input features in neural QA models, we make use of (a re-implementation of) QANet (Yu et al., 2018). By default, QANet solely uses word and character inputs. However, numerous off-the-shelf NLP tools are available that could be used to enrich these inputs with explicit linguistic information. This option is potentially interesting when trying to adapt a model to other domains: While additional training data might be expensive to obtain, these linguistic input features could boost the performance by providing a scalable, domain-agnostic source of information. We expand the per-word inputs with three different kinds of linguistic features: part-of-speech (PoS) tags, dependency relation labels and semantic roles.

PoS Tags. We hypothesized that the information about the part-of-speech of input tokens would help the neural network by reducing the number of answer candidates for specific types of questions. To extract POS tags for all contexts and questions, we used the coarse-graind PoS tag set of the spaCy library[1].

Dependency Relation Labels. We expected that syntactic information might help the model to predict the boundaries of spans with more precision. Again, we use spaCy to extract dependency information for questions and contexts. To extract dependency information per input word, we use the type label of that dependency relation in which the word is the child.

Semantic Roles. Semantic Role Labeling (SRL) deals with the problem of finding shallow semantic structure in sentences by identifying events ("predicates") and their participants ("semantic roles"). By identifying predicates and related participants and properties, SRL helps to answer "who" did "what" to "whom", "where", "when" and "how"? To do that, each constituent in a sentence is assigned a semantic role from a predifined set of roles like agent, patient or location (Márquez et al., 2008). Since semantic role labeling aims at identifying relevant aspects of events that are directly related to the above-mentioned WH questions, question answering models should directly benefit from this kinds of information.

We used the mate-plus tools (Roth and Woodsend, 2014) for parsing the complete SQuAD dataset and to obtain PropBank-labeled semantic roles per input word (Palmer et al., 2005). We added the role `<PREDICATE>` to the set of semantic roles to provide the model with pointers to the basic events. Words that did not correspond to any semantic role were assigned a `<NOROLE>` label.

Integration of Linguistic Features in QANet. In the standard QANet architecture, words and corresponding characters are embedded individually and then concatenated to obtain one representation vector per input word. Following Sennrich and Haddow (2016), we enrich this process by mapping each of the linguistic input features described above to its own embedding space and then including them into the concatenation. Figure 1 shows an updated version of the input embedding layer of QANet that includes the linguistic input features.

[1] Available at https://spacy.io/

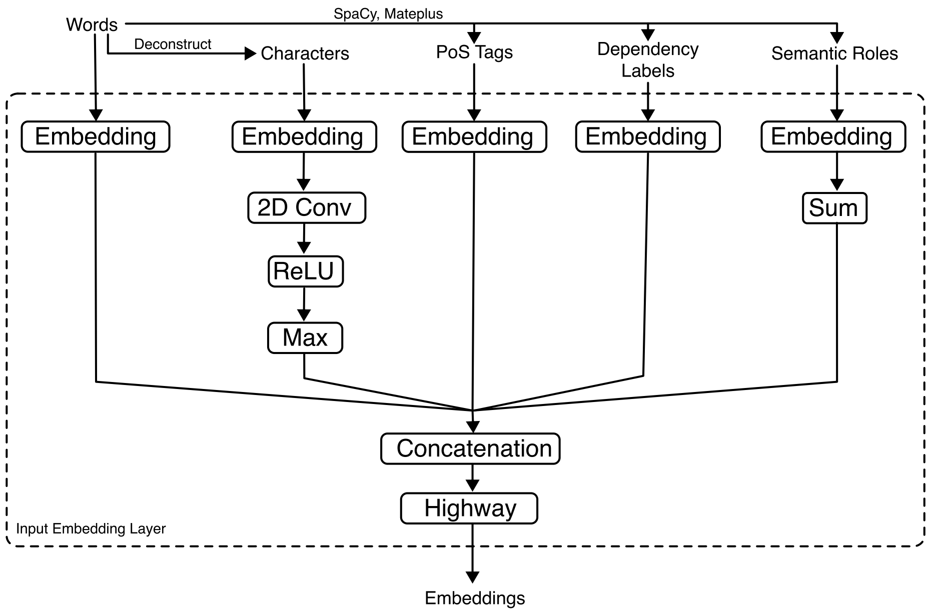

Figure 1: The low-level structure of the input embedding layer, enriched with additional linguistic inputs.

Each embedding vector consists of the embedded information of the word, its characters, its PoS tag, the label of the dependency relation in which the respective word is the child and its semantic roles. While the PoS tags and dependency relation labels are single word-level features and can be embedded by standard indexing and look-up, each word can have multiple semantic roles. Therefore, we embed each semantic role separately and aggregate over them. After preliminary experimentation with convolution, summing and taking the maximum, we decided for summing along each dimension of the semantic role embeddings[2]. This results in one aggregated semantic role embedding vector per input word. Note that we intentionally do not compute any combinations of the features mentioned above manually. We simply enrich the available word-level input information and rely on the network to find meaningful connections.

4 Experiments

To obtain a baseline, we reimplemented QANet and performed a parameter search. After that, we evaluated the integration of linguistic input features against that baseline.

4.1 Parameter Exploration in QANet

In the first experiment, we explore the effect of various parameters on open-domain QA performance in our re-implementation of QANet. The aim is to understand the impact of each parameter to compare it to the contribution of linguistic input features.

Dataset. We use the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) for parameter search. Yu et al. (2018) state that the results on development and test set are strongly correlated. Thus, improvements on the development set of SQuAD should also lead to improvements on the test set. Based on this claim, we only report results on the development set, since the test set of SQuAD is not publicly available. The training set consists of 87599 samples and the test set consists of 10570 samples. All texts are in English language.

Preprocessing. To preprocess SQuAD, we used the spaCy library for tokenization. We truncated or padded each paragraph to length 400 and each question to length 30. Each token was transformed into lower case and embedded using pre-trained GloVe (Pennington et al., 2014) embeddings. All words that were either out-of-vocabulary or not present at training time were mapped to a randomly initialized unknown token (`<UNK>`). For each token, we extracted all characters and then truncated or padded them to 16 characters per word. Each character embedding was initialized randomly.

[2]We set the maximum of semantic roles per word to 8

Parameter	ΔF1	ΔEM
word embeddings	**2.4**	1.8
character embeddings	1.6	1.7
# convolutional layers	1.5	**2.2**
shared wheights in encoding	1.3	1.3
# encoder blocks	0.9	0.9
# attention heads	0.6	1.2
# highway layers	0.4	1.0
model dimensionalty	0.5	0.8
pointwise feed-forward layers	0.2	0.0
combination of best settings	1.7	1.9

Table 1: Impact of evaluated individual parameters and the combination of their best settings on F1 and exact match (EM) scores.

Training Parameters. For regularization, we adopted the methods from Yu et al. (2018): We apply L2 weight decay on all trainable variables with $\lambda = 3 \times 10^{-7}$ and dropout on word embeddings ($p = 0.1$), as well as on character embeddings ($p = 0.05$). Additionally, a dropout of rate 0.1 is applied on every layer except from the output layer. Apart from that, we employ the stochastic depth method (Huang et al., 2016) inside each stack of encoder blocks during training. In order to compute the probabilities p_l we use a linear decay rule: $p_l = 1 - \frac{l}{L}(1 - p_L)$, following Huang et al. (2016); Yu et al. (2018). L denotes the index of the last layer in a stack of encoder blocks and the corresponding probability p_L is set to 0.9.

As an optimizer, we use ADAM (Kingma and Ba, 2014) with $\beta_1 = 0.8$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-7}$ (Yu et al., 2018). To prevent an exploding gradient, we use gradient norm clipping (Pascanu et al., 2013) with a threshold of 5. We use a learning rate warm-up schema with a logarithmic increase from 0.0 to 0.001 in the first 1000 gradient steps. After that, the learning rate is kept fixed at 0.001. Finally, we apply an exponential moving average with decay rate 0.9999 on all trainable variables. We implemented our model in PyTorch[3] (Paszke et al., 2017) and trained on a geforce GTX 1080 GPU with 12gb RAM.

Results. For evaluation, we use the F1 and exact match (EM) metrics from the official SQuAD evaluation script. See Table 1 for an overview of the impact of parameters on the scores.

[3]pytorch.org

Looking at the individual impact, the most influential parameters are the dimensionality of the inputs, namely the word and character embedding sizes. The parameter with the highest impact on F1 and the second highest impact on Exact Match is the word embedding size (ΔF1 $= 2.4$, ΔEM $= 1.8$). Surprisingly, the best setting is the smallest embedding size (50). In contrast, bigger character embedding sizes perform better than smaller ones. The best setting was a size of 300 (ΔF1 $= 1.6$, ΔEM $= 1.7$). Possibly, using no word embeddings at all and scaling up the character embeddings could increase the performance further and eliminate the need for pre-trained embeddings altogether. The most influential architectural parameter was the number of convolutional layers in model encoder blocks (ΔF1 $= 1.5$, ΔEM $= 2.2$), with the highest impact of all parameters on the exact match score when using 5 convolutional layers instead of 2. The second most influential structural parameter was sharing the weights between the embedding encoder layers for question and context (ΔF1 $= 1.3$, ΔEM $= 1.3$). The third most influential structural parameter was the number of encoder blocks in the model encoder layer, where 5 instead of 7 blocks yielded the best result (ΔF1 $= 0.9$, ΔEM $= 0.9$). The number of attention heads had a minor influence on F1 (ΔF1 $= 0.6$) but a notable influence on exact match (ΔEM $= 1.2$). Fortunately, these results are due to using only 2 attention heads. Thus, we propose to use less attention heads in order to improve results and reduce computational processing costs. The remaining parameters (the number of highway layers in the input embedding layer, the model dimensionality and the usage of pointwise feed-forward layers) all had negligible impacts on the performance. In general, we propose to set them to small values to reduce computational cost.

Combining the best settings for each parameter in isolation did not yield the best overall results in either F1 or exact match (ΔF1 $= 1.7$, ΔEM $= 1.9$). This combination was, however, the second best option for *both* F1 *and* exact match. Since this balanced quality was not apparent in any other setting, we decided to use this combination of parameters for all later experiments. The final performance of this baseline model was F1 $= 67.8$ and Exact Match $= 55.5$.

	F1	EM
Baseline	67.8	55.5
50d PoS embeddings	69.6	58.1
100d PoS embeddings	**69.7**	**58.9**
200d PoS embeddings	69.7	57.8
300d PoS embeddings	69.5	57.9
50d DL embeddings	68.6	56.6
100d DL embeddings	68.9	**57.8**
200d DL embeddings	67.8	56.6
300d DL embeddings	**69.0**	57.2
50d SRL embeddings	67.9	55.1
100d SRL embeddings	68.0	55.5
200d SRL embeddings	**68.8**	**56.6**
300d SRL embeddings	68.7	56.0

Table 2: Results of enriching the inputs with linguistic input embeddings of different sizes, in terms of F1 and exact match (EM) scores. DL refers to dependency labels, SRL to sematic role labels. No linguistic features were used in the baseline.

4.2 Impact of Linguistic Input Features

For evaluating the impact of linguistic input features, we used the same evaluation setup as for the baseline, including training data and meta-parameter choices. The embeddings for linguistic inputs were initialized randomly and then included as trainable parameters. Table 2 shows the results for varying the embedding dimensionality for each type of input feature, respectively, and also their combination.

PoS Tags. Adding PoS tags to the embedding space improves the overall performance of the network, regardless of the size of the embeddings. The best result is obtained by using PoS embeddings of size 100 ($\Delta\text{F1} = 1.9, \Delta\text{EM} = 3.4$).

Dependency Relation Labels. Enriching the inputs with dependency relation information improves the overall performance. To achieve a strong improvement, the size of the embeddings matters: While embeddings of size 200 only improve the exact match ($\Delta\text{F1} = 0.0, \Delta\text{EM} = 1.1$), all other sizes increase the F1 score as well. The biggest improvement in F1 score was achieved by an embedding size of 300 ($\Delta\text{F1} = 1.2$), while the biggest improvement in EM score was achieved by an embedding size of 100 ($\Delta\text{EM} = 2.3$).

	F1	EM
Baseline (QANet Re-Impl.)	67.8	55.5
Baseline + Linguistic Inputs	**70.5**	**60.2**

Table 3: Results of using the combination of all three linguistic input features, using the previously optimized embedding sizes (PoS and dependency tags of size 100, semantic role labels of size 200). No linguistic features were used in the baseline setting.

Semantic Role Labels. Again, the linguistic inputs improve upon the baseline performance. However, in the setting with embedding size 50, the performance slightly deteriorates. The best performance is achieved when using embeddings of size 200 ($\Delta\text{F1} = 1.0, \Delta\text{EM} = 1.1$).

Combination of all Linguistic Input Features. Table 3 shows the results for the combination of all three linguistic input features. The performance is the best in all our experiments, beating the previous baseline and individual linguistic input features ($\Delta\text{F1} = 2.7, \Delta\text{EM} = 4.7$).

5 Discussion

Due to a gap in performance between our implementation of QANet (cf. Table 3) and the results from the original paper[4], we are not able to tell whether our optimized parameters are specific to our settings or should be preferred in general. The mismatch in performance could be due to various implementational differences (such as using PyTorch instead of Tensorflow), variation in preprocessing (e.g. tokenization with spaCy instead of NLTK) or the training procedure (such as trainable word embeddings). Still, we consider the relative improvements due to linguistic inputs compared to our baseline to be very insightful and promising.

Overall, each individual linguistic input achieved a small improvement of the performance. The best performing feature were the PoS embeddings, with the biggest improvements in both F1 and exact match ($\Delta\text{F1} = 1.9, \Delta\text{EM} = 3.4$). The second best feature were the dependency relation labels ($\Delta\text{F1} = 1.1, \Delta\text{EM} = 2.3$), followed by semantic role labels ($\Delta\text{F1} = 1.0, \Delta\text{EM} = 1.1$). Combining all linguistic input features led to even better results ($\Delta\text{F1} = 2.7, \Delta\text{EM} = 4.7$). This indicates that although the PoS tag embeddings

[4]F1=82.7, EM=73.6, as reported by Yu et al. (2018).

have the strongest impact on the performance, dependency relation and semantic role embeddings still provide useful additional information.

Importantly, this also shows that adding linguistic features into a model that incorporates optimally selected hyperparameters yields an additional performance benefit, suggesting that linguistic input features have a bigger impact on model performance than hyperparameter optimization alone. These findings underline the usefulness of linguistic features as a simple and readily available tool for enhancing the performance of QA models. Contrary to our intutions, the increase in performance seems to be lower in those features that are higher up in the hierarchy of linguistic abstraction: PoS tags, which merely provide a shallow and coarse-grained approximation of meaning, perform better than semantic roles, which provide a lot of information that should support the question answering task. This could be explained as follows: First, PoS tags (and dependency relation labels) are available for every word in a sentence, but only some words correspond to semantic roles. This sparsity renders this input feature less reliable. Second, our current aggregation approach towards semantic role embeddings might not be optimal.

The results suggest that syntactic information in dependency labels might help the model to find more precise boundaries: While the F1 score only increased by 1.1, the exact match was improved by 2.3. Even though the employed embedding process for the dependency labels was rather simple and not tailored to the underlying dependency tree structure, the feature was useful to the network. Probably, the performance of this feature could be increased either by classical feature engineering to construct more specific information from the dependency tree or by using a more suitable network architecture for embedding such structures.

Feature Engineering for Neural Architectures At least in higher-level tasks like QA, we observe a recent trend in the literature to focus on improving model architecture over improving input features, possibly due to the ability of neural networks to learn hierarchical features. Following this paradigm, the state-of-the-art in many tasks has recently been improved, for example in semantic role labeling through a deep highway BiLSTM (He et al., 2017). However, the last papers that make use of semantic roles are mostly from the last decade (Shen and Lapata, 2007; Kaisser and Webber, 2007; Sammons et al., 2009; Wu and Fung, 2009; Liu, 2009; Gao and Vogel, 2011). Most current QA architectures use word and character embeddings only (Seo et al., 2016; Wang et al., 2017; Xiong et al., 2016; Shen et al., 2017; Hu et al., 2018; Yu et al., 2018).

Our results suggest that the QANet model benefits more from better inputs than from optimizing its structure: When exploring the parameters, we observed that the embedding dimensionality of words and characters had the biggest impact. In the experiments regarding linguistic input features, we found that injecting linguistic information into the input had a stronger effect than any of the previously explored parameters.

Based on this, it might be worthwhile to invest more work into the investigation of the type of inputs one feeds into a neural model. An advantage of better inputs is their adaptability, since input features can possibly be used off-the-shelf for a wide range of tasks, while architectures typically have to be fine tuned on supervised training data. Another advantage might be that empirical gains are better interpretable: In our experiments, linguistic input features had a bigger impact on exact match (finding exact boundaries of the correct answer) than on F1 (overlap with correct answer). A thorough investigation of this intuitive link between input and output qualities would be necessary to support this claim.

6 Conclusion

This paper addresses the question as to whether neural QA models benefit from linguistic input features at different levels of abstraction in terms of their presumed generalization capacities across domains. To this end, we evaluate the impact of injecting different linguistic inputs (PoS tags, syntactic dependencies, semantic roles) into a deep neural QA architecture on open-domain performance over SQuAD, which constitutes the largest open-domain benchmark data set currently available.

In our experiments, the highest individual performance gain was achieved by adding PoS tags to the input representation, but a combination of all evaluated linguistic features led to the best results overall. We noticed that linguistic input features had a bigger impact on the exact match score than on the F1 score. Thus, we hypothesize that the linguistic information facilitates boundary detection, while locating answer candidates in general may largely depend on word-level semantics.

In future work, it might be instructive to explore how well linguistic features perform without word or character inputs. This would give a better basic intuition about the performance of each individual feature. Another interesting research direction could be to investigate further linguistic information like lemmatized words, subword tags and morphological features (Sennrich and Haddow, 2016), named entity recognition and distance and position features. Apart from that, novel approaches for integrating this information could be worthwhile: While features that have one value per word (like PoS tags) can be easily embedded, relational features or features with multiple values per word (like semantic roles) need some kind of aggregation. While we chose summing over individual dimensions, various other approaches like convolution, recurrent layers or even self-attention might lead to better results. Tree-structured features like dependency trees might benefit from recursive encoding layers (Socher et al., 2011).

We conclude that linguistic input features provide meaningful information to neural QA models and that they improve performance on a general domain dataset. In future, we plan to evaluate whether they might be employed for generalizing QA models to new specialized domains.

Acknowledgments

This work was supported in part by the H2020 project Prêt-à-LLOD under Grant Agreement number 825182.

References

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 740–750.

Qin Gao and Stephan Vogel. 2011. Utilizing target-side semantic role labels to assist hierarchical phrase-based machine translation. In *Proceedings of Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation, SSST@ACL 2011, Portland, Oregon, USA, 23 June, 2011*, pages 107–115.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 473–483.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4099–4106.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 646–661.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *CoRR*, abs/1711.07341.

Michael Kaisser and Bonnie Webber. 2007. Question answering based on semantic roles. In *Proceedings of the workshop on deep linguistic processing*, pages 41–48. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 595–603.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1378–1387, New York, New York, USA. PMLR.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Lluís Márquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: An introduction to the special issue. *Computational Linguistics*, 34(2):145–159.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 407–413.

Mark Sammons, V. G. Vinod Vydiswaran, Tim Vieira, Nikhil Johri, Ming-Wei Chang, Dan Goldwasser, Vivek Srikumar, Gourab Kundu, Yuancheng Tu, Kevin Small, Joshua Rule, Quang Do, and Dan Roth. 2009. Relation alignment for textual entailment recognition. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 83–91.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Dan Shen and Mirella Lapata. 2007. 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*, pages 12–21.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1047–1055.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 129–136.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 189–198.

Dekai Wu and Pascale Fung. 2009. Semantic roles for SMT: A hybrid two-pass model. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA, Short Papers*, pages 13–16.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics* *and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1127–1137.