Exploring the effects of semantic tag augmentations for SQuAD

Stanford CS224N Default Project
Track: SQuAD

Narvin Phouksouvath
Department of Computer Science
Stanford University
narpho23@stanford.edu

Jonathan Tseng
Department of Computer Science
Stanford University
jtseng20@stanford.edu

Abstract

Question-Answering is a broadly researched NLP problem which has seen many contributions. Many publications that approach the QA task leverage an attention mechanism of some kind. In our paper, we explore the effectiveness of augmenting three different models which use attention with character-level and semantic embeddings, and examine the qualitative effects on their attention behaviors.

1 Key Information to include

- TA mentor: Michi Yasunaga
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Reading comprehension is one of the fundamental problems of NLP. One of the ways we measure a machine's ability to comprehend text is by seeing its performance on question answering tasks. Question answering (QA) is a heavily researched topic and has seen a number of breakthroughs with recent advancements in deep learning, most important of which may be the transformer. Question Answering tasks come in many different forms, depending on what dataset researchers use to train and evaluate a model on. One of these datasets is the Stanford Question Answering Dataset (SQuAD). SQuAD provides crowdsourced questions based on passages sourced from Wikipedia, with answers being a segment of text, or span, from the context passages. The best model in the official SQuAD leaderboard reaches an F1 score of 93.214, beating even human performance.

Our paper focuses on three different models that were trained and evaluated on SQuAD: Bi-directional Attention Flow (BiDAF), R-NET, and QANet. Each of these models have demonstrated strong performance on SQuAD, but have distinct approaches to decoding texts. SQuAD's latest iteration SQuAD 2.0 introduces a new challenge for these models: to abstain from answering when no answer can be found from the provided context passage. Because the models of focus in this paper were created before SQuAD 2.0, the performance of these models on the newer dataset have, as anticipated, decreased.

In our paper, we reimplemented these models and trained them on the SQuAD 2.0 dataset. In addition, we investigated how the performance of these models changes with added features, such as character-level embeddings, part of speech (POS) tagging, and named entity recognition (NER) tagging. Additionally, we provide a qualitative examination of how these modifications changes the attention mechanism that underlies each of these three models.

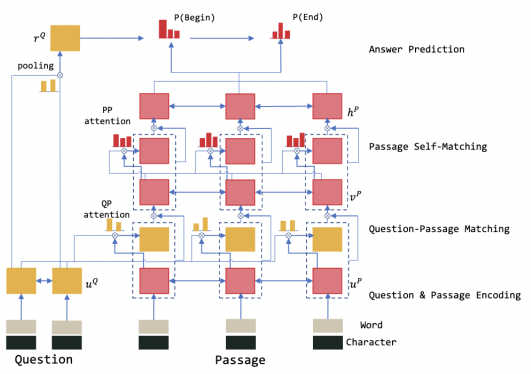


Figure 1: R-NET Architecture

3 Related Work

Models have long leveraged attention as a means of tackling the SQuAD challenge. BiDAF introduced the notion of context-query attention to SQuAD, allowing it to augment prior state-of-the-art recurrent models, and score competitively on the leaderboard at the time. Soon afterward, researchers innovated on the attention mechanism with R-NET, which contributes two significant technical details: first, a modification of attention units that additionally gates the output in a manner similar to the gates found in GRU and LSTM. This gated attention mechanism is used in R-NET in place of vanilla additive attention. The second key contribution is the introduction of a "self-attention" mechanism. The authors of R-NET claim that QA tasks require understanding of question-to-passage correspondence, provided by the gated attention units, but they might also require correspondence between different parts of the passage itself, in cases where the answer might have clues derived from different pieces of the passage. The authors note that while a typical recurrent model can theoretically remember all of its input, and therefore derive information from past contexts, in practice this is not the case on long horizons. To achieve contextualization, R-NET also attends over its own intermediate alignment layers. In theory, self-attention encourages easier correspondence between different sections of the passage, especially for longer inputs, but can pose prohibitive computational load when attending very long passages against themselves.

Since the publication of R-NET, purely attention-based models like transformers have dramatically changed the landscape of deep learning by providing a cost-efficient yet effective alternative to RNNs [1, 2]. Many QA models have embraced transformers; among the first successful examples of these is QANet. QANet does away with the RNNs that R-NET used heavily, instead encoding texts using a transformer-like module, detailed in Section 4.3.1.

4 Approach

Our experiments are built on three basic model architectures, each coded from scratch in PyTorch: BiDAF, R-NET, and QANet. We extend these models with additional information in the form of character-level embeddings, part-of-speech embeddings, and named-entity embeddings, and present qualitative and quantitative analyses of the effects of these extensions.

4.1 BiDAF

The baseline model, provided by the starter code, is an implementation of BiDAF [3]. Pre-trained word embeddings first go through a Contextual Embedding Layer, which uses LSTMs to gather information from surrounding words in the context passage. The next layer, which is the key layer of BiDAF, is the Attention Flow Layer, which uses an bidirectional attention mechanism, letting attention flow from context to question and question to context. The embeddings then go through a Modeling Layer, which uses bidirectional LSTMs to encode the query-aware representations of context words. We used this model (without character embeddings) as a benchmark for the performance of our other models.

4.2 R-Net

Following the architecture of R-NET [4], our model is divided into 5 layers: the embedding layer, the encoding layer, the question-passage matching layer, the passage self-matching layer, and the output layer.
In the embedding layer, we create embeddings for both the question and context passage, adapting the Embedding layer from the starter code to produce word and character-level embeddings. We loosely follow the description provided in R-NET to create a character embdedding that we can concatenate to each word embedding. We run the embeddings obtained from the provided character vectors through a bidirectional GRU, taking the final hidden state and concatenating it to our word embeddings [4]. We additionally project the embeddings to have a dimensionality of the hidden size, then apply a highway network [5] to refine the embeddings, a feature used in BIDAF, but not in R-NET.
To encode the context and question, we run the context and question embeddings through an RNN [4]; in our implementation, we use GRU. The motivation for using GRU as opposed LSTM is GRU's computational efficiency.

We then run our both our encoded outputs through a gated attention-based recurrent network [4], which matches the question with the passage, and outputs a sentence-pair representation $v_t^P = \text{RNN}(v_{t-1}^P, c_t)$, where $c_t = att(u^Q, u_t^P)$ is an attention-pooling vector of the question obtained by:

$$s_j^t = \mathrm{v}^\top \tanh(W_u^Q u_j^Q + W_u^P u_t^P + W_v^P v_{t-1}^P) \tag{1}$$

$$a_i^t = \text{softmax}(s_i^t) \tag{2}$$

$$c_t = \sum_i a_i^t u_i^Q \tag{3}$$

The layer uses another gate to determine which parts of the context passage are relevant to the question and attend to them.

$$g_t = \text{sigmoid}(W_g[u_t^P, c_t]) \tag{4}$$

$$[u_t^P, c_t]^* = g_t \odot [u_t^P, c_t] \tag{5}$$

$[u_t^P, c_t]^*$ is used in subsequent calculations instead of $[u_t^P, c_t]$

The self-matching attention layer directly matches the question-aware passage representation we obtained from the previous layer on itself. The implementation for this layer is similar to the question-passage matching layer before it, except we attend the v^P embeddings against each other:

$$c_t = att(v^P, v_t^P) \tag{6}$$

In the output layer, we use a pointer network [6] to select a start and end position in the passage to return an answer. We obtain these positions by using the same attention mechanism used before

$$s_j^t = \mathrm{v}^\top \tanh(W_h^P h_j^P + W_h^a h_{t-1}^a) \tag{7}$$

where h_{t-1}^a represents the last hidden state of the pointer network. We use a attention-pooling vector based on current predicted probability a^t to obtain the next hidden state. The initial state of the pointer network is obtained by using the attention mechanism on the question embeddings and the output from the previous layer.

$$r^Q = att(u^Q, V_r^Q) \tag{8}$$

We coded each layer from the bottom up, using basic Pytorch functions, though the embedding layer is built over the vanilla embedding layer in the starter code. For the RNNs, we used GRU, as was done in the R-NET paper [4]. The baseline for the BiDAF model was implemented entirely in the starter code.

4.3 QANet

QANet is a development on both the BiDAF and R-NET models, which shares the Attention Flow layers and the character-level embeddings, but replaces all recurrent encoders with "Encoder Blocks" [7].

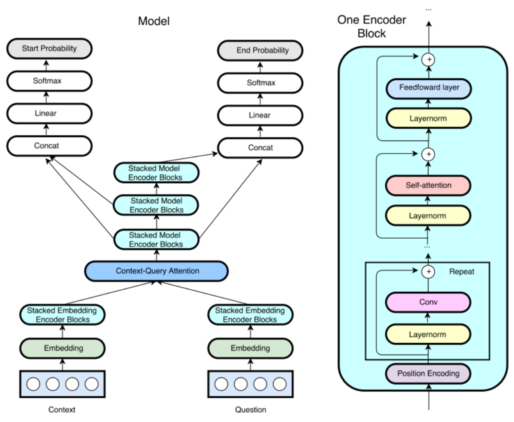


Figure 2: The QANet architecture, with the encoder block shown at right.

4.3.1 Encoder Block

The Encoder Block, seen at right in Figure 2 uses positional encodings, followed by depthwise-separable convolutions [8], a self-attention layer, and a fully-connected output projection. To normalize inputs and improve training, every layer is preceded by layer normalization [9], uses a residual connection, and uses dropout during training. The Encoder Block replaces both the encoding layer and the modeling layer. In our implementation, we use a single block of 4 convolution layers with kernel size of 7 in the encoding layer, and 3 stacks of 7 blocks, each containing 2 convolution layers of kernel size 5. The primary advantage of the Encoder Block is that, in eliminating the dependence on recurrent units, training speed is increased significantly, which allows a corresponding increase in model complexity.

4.3.2 Context Query Attention

As mentioned above, QANet shares the BiDAF attention mechanism, detailed earlier in this paper.

4.3.3 Output Layer

QANet takes M_1, M_2, M_3, the outputs of the three encoder stacks that comprise the modeling layer. It models the start and end of the answer span as:

$$p_{start} = \text{softmax}(W_0([M_1; M_2]))$$

$$p_{end} = \text{softmax}(W_0([M_2; M_3]))$$

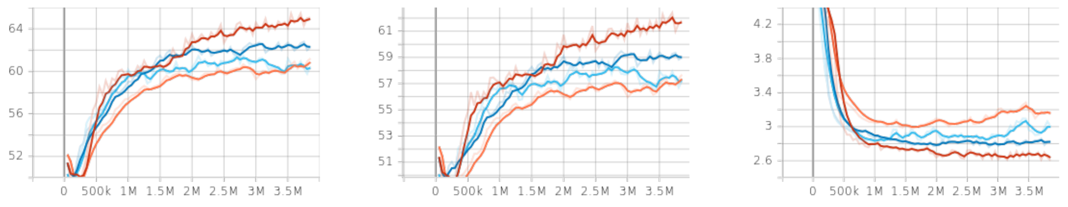
5 Experiments

5.1 Character-level Embeddings

The first of our experimental extensions is a character-level embedding for BiDAF. R-NET and QANet implement a character embedding by default, so no change is made in those cases. We chose to implement character embeddings as the final hidden state of a bi-directional GRU applied to the characters in each word token, as seen in R-NET.

5.2 POS and NER Tagging

To add additional features to our embeddings, we incorporated part of speech (POS) tagging and named entity recognition (NER) tagging. POS tagging involves labeling each token in a document sentence based on its part of speech (e.g. noun, verb, adjective) or grammatical function (e.g. symbol, punctuation). POS tags can be further subdivided into smaller tag categories, such as making distinctions between common nouns and pronouns, or creating separate tags for different verb tenses. NER tagging labels each token based on its named entity type. Named entity recognition is meant to



(a) F1 (b) EM (c) NLL

Figure 3: Training scores of BiDAF (orange), BiDAF-POS (cyan), R-NET (blue), and QANet (red)

identify and distinguish specific details in a text. A named entity includes anything that has a proper name or specific value; this can include persons (e.g. Leland Stanford), organizations (e.g. NATO), places (e.g. Kyoto), and dates (e.g. 1871). Words that are not named entitites (e.g. the, house, happy) are tagged as non-entities. The spaCy taggers have 18 entity types.

While tokens tagged by humans would ensure accuracy, human tagging is costly both in money and time and is not within the scope of our project. Instead, we opted to tag our data using NLP pre-trained taggers provided by Natural Language Tookit (NLTK) and spaCy. These taggers can potentially add rich descriptors of the behavior of individual words and can augment a model's understanding of its interactions with other words, especially early on in training.

We incorporated tags by using an embedding layer for each of 36 POS types and 18 entity types, including an "unknown" embedding, and concatenating the embedding prior to the encoder, which is a Highway Encoder [5] in BiDAF and QANet, and a GRU encoder in R-NET.

6 Experiments

6.1 Data

We trained and tested our model on the SQuAD 2.0 dataset, which contains passages from Wikipedia and crowdsourced questions and answers based on those passages. We use the GloVE embeddings [10] from the provided setup in the starter code.

6.2 Evaluation method

We measured the performance of our model using two different metrics: the Exact Match (EM) score and the F1 score. Final scores are obtained by averaging EM and F1 scores across the entire evaluation dataset.

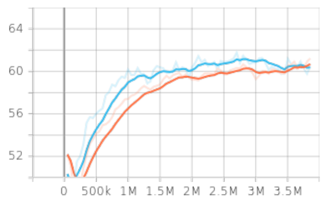
6.3 Experimental details

The BiDAF and R-NET models were optimized using AdaDelta [11] with a learning rate of 0.5 and a dropout chance of 20%. QANet was trained using Adam [12] with a learning rate of 0.001 and a dropout chance of 10%. All models were trained to convergence in at most 30 epochs, and evaluated on the dev set every 50000 steps; the highest-scoring checkpoint is the one we report.

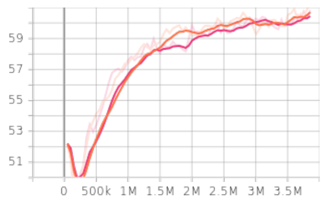
6.4 Results

We quantitatively evaluate the success of our model using F1 and EM scores, reported in Table 1. The addition of character embeddings to BiDAF increases performance noticeably, and allows it to more closely match the performance of R-NET and QANet. Adding part of speech information to our embeddings had a smaller but still noticeable improvement on BiDAF's performance. NER tagging worsened performance universally, and as such, were dropped from our experiments with the larger models. Further details of NER are discussed in Section 7.2.

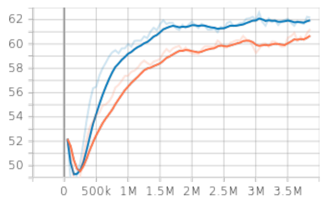
Augmenting our implementation of R-NET using POS tags improved performance to a similar degree. An inspection of the F1 curves of the two R-NET models shows that R-NET POS improves slightly more at all times. This indicates that POS tags may provide slightly more expressive input embedding



(a) Vanilla (orange), POS (cyan)

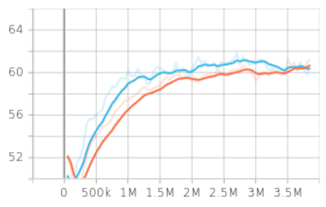


(b) Vanilla (orange), NER (pink)

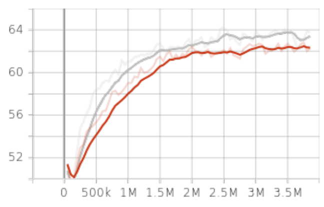


(c) Vanilla (orange), Char (blue)

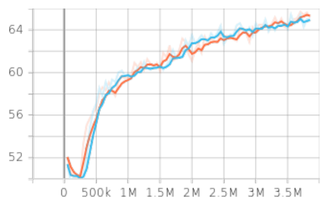
Figure 4: F1 training scores of BiDAF models



(a) BiDAF: Vanilla (orange), POS (cyan)



(b) R-NET: Vanilla (red), POS (grey)



(c) QANet: Vanilla (cyan), POS (orange)

Figure 5: Impact of POS tagging on F1 training scores

that improves gradients. Notably, POS improved R-NET more than it improved BiDAF, which might be because we did not modify the BiDAF model size to accommodate a more expressive input.

The addition of POS tags improves the performance of QANet by a slim margin, where an examination of the progression of EM and F1 scores shows that the two are so close that the improvement can be characterized as due to randomness. We hypothesize that this is due to the significantly increased model complexity of QANet – since vanilla QANet is complex and able to learn rich representations of words already, the imprecise nature of POS tags likely ends up not contributing to the performance, as the model learns to ignore it.

It is important to note in our analyses that POS tags may be imprecise, causing exact effects to be hard to diagnose, as discussed in greater detail in Section 7.1.

Among all the models, the two versions of QANet perform best, which is not surprising since it is the most complex model. We note that the performance on the test set is noticeably worse than on the dev set. We believe that this is due to a difference in distribution between the two sets, as well as some overfitting to the dev set, as the tested model is chosen based on which performs best on dev set evaluation.

Performance on Dev Set			
Model	F1	EM	AvNA
BiDAF Baseline	61.22	57.69	68.06
BiDAF POS	61.76	58.63	67.94
BiDAF NER	60.83	57.52	67.92
BiDAF NER POS	61.75	58.44	68.21
BiDAF Char Embeddings	62.59	59.17	68.96
BiDAF Char Embeddings POS	63.14	59.94	69.8
R-NET	62.66	59.47	69.27
R-NET POS	64.23	60.83	70.71
QANet	65.45	**62.41**	71.55
QANet POS (dev)	**65.62**	**62.41**	**71.79**
QANet POS (test)	62.88	59.73	-

Table 1: Quantitative results of tested models

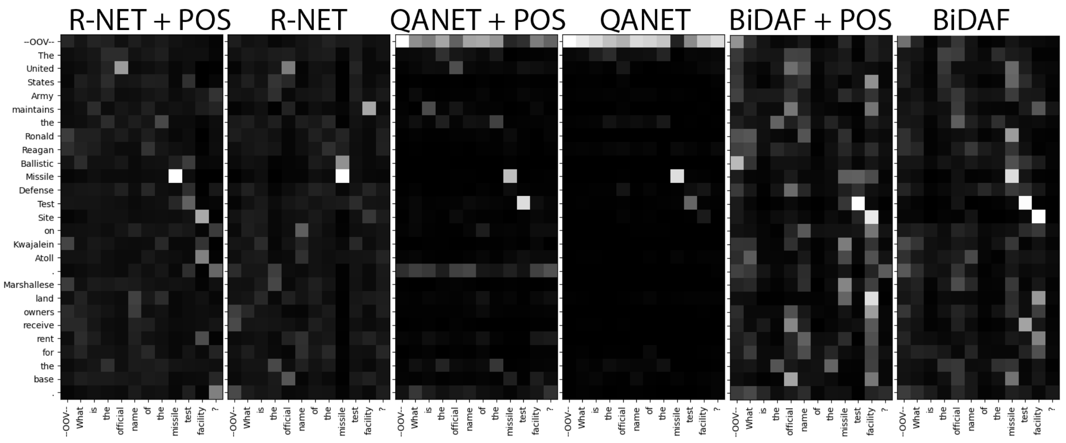


Figure 6: A visualization of the attention layers of each architecture, with and without POS tags. BiDAF is significantly impacted by POS, while QANet is almost unaffected.

7 Qualitative Analysis and Discussion

In addition to our observation of the performance metrics and the trends therein, we believe that examinations of the model attention layers help us gain an understanding of their inner understandings of natural language. Shown in Figure 6 is a visualization of the context-query attention of each of the three models, with and without POS tags – the attention maps produced by question answering models can be used as rough visual aids to understand their inner “understanding" of natural language. Although it is generally not possible to use attention maps in a fully-interpretable way, the high-level effect of POS tags on the three models is clearly visible. The effect of POS tags is significant on BiDAF, less so for R-NET, and almost nonexistent for QANet. This confirms our hypothesis that these more complex attention-based models learn rich representations of words that, over the course of training, incorporate semantic understanding of the words such as part-of-speech features. By comparison, BiDAF is significantly affected by POS tags, both qualitatively and quantitatively, which suggests that its understanding of words is limited and does not yet reach full semantic understanding.

It is also possible that the addition of POS tags was beneficial to R-NET and BiDAF not for semantic reasons, but as a side effect of its noisy nature. As discussed in Section 7.1, POS tags are slightly imperfect, which allows it to potentially act as a regularizer that prevents overfitting. This theory is supported by the training progressions we observed: Figure 4 shows that the dev set loss of the BiDAF model exhibits clear overfitting towards later iterations, while BiDAF POS shows a better fit on the dev distribution.

As discussed in Section 7.2, NER tags are similarly noisy. However, our experiments showed that NER tags universally worsened performance. Although time constraints caused us to cut short experimentation on the addition of NER to the more complex models, its failure to benefit BiDAF indicated that it was unlikely to improve BiDAF's more complex counterparts.

The immediate lack of usefulness of external NER tags is visible from observing the text summaries from training the baseline model. We found that even an unmodified BiDAF model, the lowest-performing of all the models we tested, did not have difficulty identifying named entities (notice BiDAF attends to “Ronald Reagan" in the figure). While some questions have named entities as their answers, QA models demonstrate an ability to make these answer choices based on context as opposed to an explicit knowledge of whether or not a word is a named entity. Furthermore, the most difficult questions in SQuAD require advanced information synthesis, whereas NER can, at best, help only with simple name retrieval, a category that the baseline has no trouble with.

7.1 Limitations of POS Tagging

Because NLTK and spaCy taggers are pretrained on separate corpora and use statistical models, they can be subject to imprecision in their classification, which limits their usefulness in a deep learning context. Consider the question and context from SQuAD:

Context: The mermaid (syrenka) is Warsaw's symbol and can be found on statues throughout the city and on the city's coat of arms. This imagery has been in use since at least the mid-14th century. The oldest existing armed seal of Warsaw is from the year 1390, consisting of a round seal bordered with the Latin inscription Sigilium Civitatis Varsoviensis (Seal of the city of Warsaw). City records as far back as 1609 document the use of a crude form of a sea monster with a female upper body and holding a sword in its claws. In 1653 the poet Zygmunt Laukowski asks the question:
Question: What is polish for "mermaid"?.

The spaCy tagger outputs the following:

Context (truncated): `[('The', 'DET'), ('mermaid', 'ADJ'), ('(', 'PUNCT'), ('syrenka', 'NOUN'), (')', 'PUNCT'), ('is', 'VERB'), ('Warsaw', 'PROPN'), ("'s", 'PART'), ('symbol', 'NOUN'), ..., (':', 'PUNCT')]`

Question: `"[('What', 'NOUN'), ('is', 'VERB'), ('polish', 'ADJ'), ('for', 'ADP'), ('"', 'PUNCT'), ('mermaid', 'VERB'), ('"', 'PUNCT'), ('?', 'PUNCT')]"`

Notice that in both context and question, the tagger mislabels "mermaid". Moreover, the tagger is not consistent with how it tags "mermaid"; in the context, it is labeled as an adjective, while in the question, it is labeled as a verb. This presents a major problem for the model, as "mermaid" is a key word that the model has to recognize in both the question and the context in order to answer the question. By not only assigning the wrong part of speech to "mermaid", but also not assigning the same part of speech in both question and context, we introduce inaccuracies to the input embeddings that could compromise the model's ability to identify important information. A model exposed to overly imprecise tags will eventually learn to ignore them. However, the shown example is an edge case ("mermaid" is likely a rare out-of-distribution word), and empirically, the POS taggers that we use have been shown to achieve F1 scores above 90% on the NLTK Brown corpus, indicating that the inaccuracy is likely not so significant that it would damage a model that uses them.

7.2 Limitations of NER Tagging

NER taggers experience the same limitations as POS taggers: they rely on pre-trained models that are not entirely accurate. However, spaCy NER taggers have a lower accuracy than POS taggers (97.2 vs 85.5 on `en_core_web_lg` pipeline), which amplifies its problems. In addition, identifying named entities have little importance in questions that don't ask about named entities (e.g. "How do centripetal forces act in relation to vectors of velocity?").

8 Conclusion

We explored the effects of different embeddings on attention behavior in three models. BiDAF's dramatic improvement from the use of character-level embeddings underscore the importance of including morphological information in reading comprehension tasks. From our investigations of POS and NER tagging, we found that POS tagging could improve older recurrent models, but NER tagging was detrimental to performance. In the case of POS tags, their noisy nature prevented overfitting and provided improvements to performance, but more complex models eventually learned them away in later training. This suggests that a strategic tuning of other regularization techniques (e.g. dropout, normalization) is likely altogether enough to replace POS tags. Using more powerful and accurate state-of-the-art taggers may improve the performance seen in this paper, but it is very unlikely that it would make a significant impact given the diminishing returns observed on QANet. We concluded that tagging features are ultimately unnecessary in self-attention models, adding superfluous information to sophisticated models that can already infer these complex grammatical properties. While both QANet and R-NET had impressive results, QANet's superior performance and computational efficiency suggest RNNs have a diminished importance in NLP problems such as question-answering. Indeed, purely attention-based models, such as BERT [2], dominate the state-of-the-art in all major NLP tasks.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[4] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. May 2017.

[5] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015.

[6] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer, 2016.

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

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[9] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[11] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.