LTAT.01.001 Course Project: AllenNLP BiDAF Machine Comprehension Model and Modifications

Elizaveta Korotkova
Institute of Computer Science
University of Tartu
elizaveta.korotkova@gmail.com

Abstract

This project explores the Bi-Directional Attention Flow model for the task of machine comprehension as it is re-implemented in the AllenNLP framework. I retrained and evaluated the original model and made some modifications to it, such as using different pre-trained word embeddings and no pre-trained embeddings at all, adding explicit part of speech and named entity recognition tags and dependency labels, and using an LSTM character encoder. These modifications mostly improved the model's performance, but not by much, with contextualized embeddings leading to the largest improvement.

1 Background

1.1 Machine Comprehension

The task of machine comprehension entails automatically answering a query about a given context paragraph. It is distinct from general question answering in that the answer is not to *any* question, but to one the answer to which can be inferred from the specific text the machine is presented with.

An example of how a machine comprehension system should work could be the following:

Paragraph: A rainbow is a meteorological phenomenon that is caused by **reflection, refraction and dispersion of light in water droplets** resulting in a spectrum of light appearing in the sky. It takes the form of a multicoloured circular arc. Rainbows caused by sunlight always appear in the section of sky directly opposite the sun.[1]
Query: What causes rainbows to appear?
Answer: reflection, refraction and dispersion of light in water droplets

The paragraph contains the answer to the question, and the machine comprehension system should be able to choose the appropriate span of the paragraph as its output.

Recently, the emergence of the neural attention mechanism (Bahdanau et al., 2015), initially designed for use in machine translation, has brought advancement in the machine comprehension task as well. Attention allows the models to focus on a specific area of the paragraph that is most relevant to the question.

1.2 SQuAD Dataset

The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a commonly used dataset for training and evaluation of machine reading comprehension systems. It is based on 536 Wikipedia articles on various topics. From the selected articles, more than 23K separate paragraphs were extracted. Crowdworkers were employed to create questions based on these paragraphs, the answers to which are spans of the source texts. The eventual dataset consists of more than 100K such questions and answers. SQuAD was used for training and evaluation of the model I experiment with, described in the following section.

1.3 BiDAF Model

The Bidirectional Attention Flow for Machine Comprehension (BiDAF) model (Seo et al., 2016) is a widely used baseline which uses the attention mechanism. It achieved state-of-the-art accuracies on the SQuAD dataset in 2017.[2]

BiDAF is a hierarchical multi-stage architecture. It models the context paragraph at different levels of granularity, using character-level, word-level and contextual embeddings. Attention mech-

[1] https://en.wikipedia.org/wiki/Rainbow

[2] https://rajpurkar.github.io/SQuAD-explorer/

anisms are used in both directions, such that the query is represented with attention to context, and the context is represented with attention to query, and they can provide situation-aware complimentary information to each other.

2 Experiments

This section describes my experiments with the AllenNLP BiDAF model.[3] I retrained the original model and experimented with different word embeddings, adding explicit tags to the input, and changing the architecture of character-level encoder. For quick comparison of scores of all models on the SQuAD development set, see Table 1.

Table 1: EM and F1 scores of all models on the SQuAD development set

Modifications	EM	F1
Original	68.3	77.8
Original, retrained	67.3	76.9
ELMo embeddings	**70.9**	**80.3**
Trainable embeddings	57.8	68.4
With POS tags	68.1	77.8
With NER tags	67.4	77.2
With dependency labels	67.6	77.4
LSTM character encoding	66.9	76.4

2.1 Original Model

The original model I further build on is the BiDAF model re-implemented in the AllenNLP library[4] (Gardner et al., 2018). It achieves EM (exact match) score of **68.3** on the SQuAD development set and F1 score of **77.8**. (Exact match means that the predicted span of the context paragraph matches the human-annotated span precisely. F1 is a more permissive score: it calculates the harmonic mean of precision and recall of the predicted span on character level.)

Let us see how well this model fares when presented with the query about rainbows:

Paragraph: A rainbow is a meteorological phenomenon that is caused by reflection, refraction and dispersion of light in water droplets resulting in a spectrum of light appearing in the sky. It takes the form of a multicoloured circular arc. Rainbows caused by **sunlight** always appear in the section of sky directly opposite the sun.
Query: What causes rainbows to appear?
Answer: sunlight

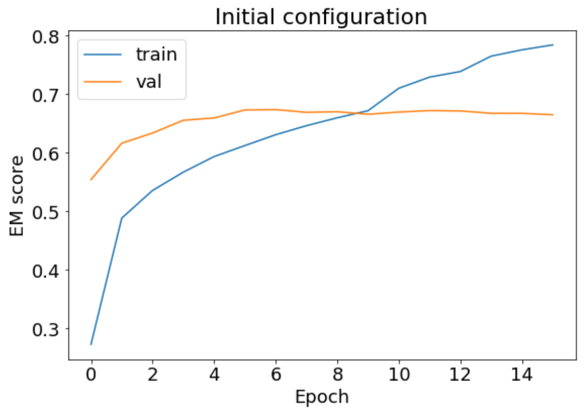

Figure 1: Training progress of the original AllenNLP BiDAF model, retrained. The figure shows the exact match (EM) score on training and validation sets during training.

The answer is not the one I would give, but it is understandable why the system could make this mistake. Even though not all rainbows are caused by sunlight, so the answer is not strictly true, it is close to reality.

However, as this model was probably carefully selected as the best performing of multiple random initializations, I retrained a model of exactly the same configuration[5] to have an idea of what difference a single run would make. My retrained model showed EM development score of **67.3** and F1 score **76.9**, both around one point lower than the model provided by AllenNLP. That is the result I compare further results to. Training this model took around 3.5 hours on a falcon1 HPC GPU (NVIDIA Tesla P100 with 16GB of VRAM). The training progress is shown in Figure 1.

2.2 Word Embeddings

2.2.1 ELMo

The original model uses pre-trained GloVe word embeddings (Pennington et al., 2014). As we have seen previously (in the homeworks of this course) that contextualized word representations can improve model performance on various tasks, I tried

[3]Code: https://github.com/lisskor/nlp_course/tree/master/project
[4]https://allennlp.org/models

[5]https://github.com/allenai/allennlp/blob/master/training_config/bidaf.jsonnet

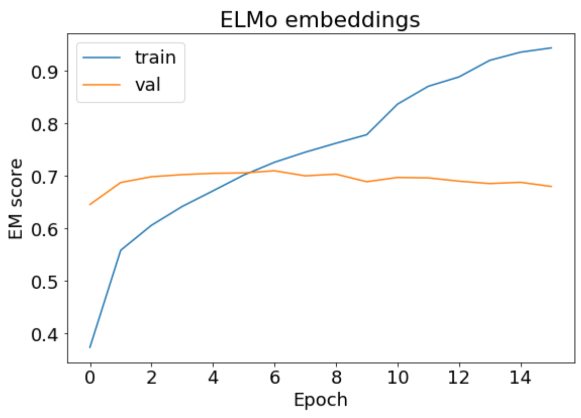

Figure 2: Training progress of the AllenNLP BiDAF model with ELMo embeddings. The figure shows the exact match (EM) score on training and validation sets during training.

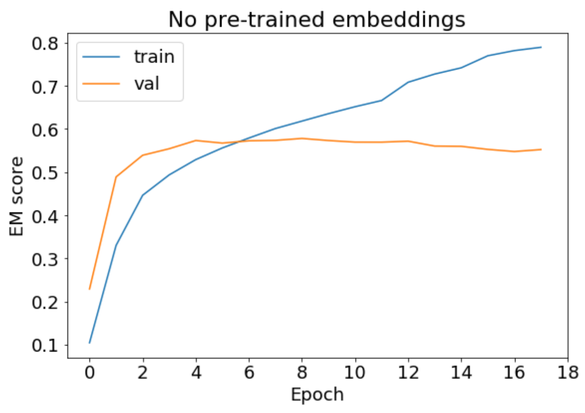

Figure 3: Training progress of the AllenNLP BiDAF model with no pre-trained embeddings. The figure shows the exact match (EM) score on training and validation sets during training.

using ELMo embeddings (Peters et al., 2018) instead.

The model with ELMo embeddings achieved EM score on the validation set **70.9**, and F1 score **80.3**. (Training progress shown in Figure 2.) That is an improvement of more than 3 points on each score compared to the original retrained model.

However, training the model took about 11.5 hours on a similar falcon1 HPC GPU (more than 3 times slower than the original model with GloVe embeddings). Therefore, I decided not to experiment with this configuration further for the sake of a speed-up, and implemented further changes on top of the initial model with GloVe embeddings. I assume that using ELMo embeddings could improve results of other experiments as well.

2.2.2 Trainable Embeddings

I also tried not using any pre-trained embeddings at all to see whether the model can learn useful embeddings on its own during training for the target task. I used trainable embeddings of dimension 100 (the same size as the initially used GloVe embeddings).

This model only achieved **57.8** EM score and **68.4** F1 score on the validation set. That is a big drop compared to the original configuration. See learning curves in Figure 3.

2.3 Tagging

Authors of the SQuAD dataset (Rajpurkar et al., 2016) report that more than 50% percent of answers in the development set are proper nouns and numbers, including dates. More than 30% are common noun phrases, and only around 15% belong to other types. They also show that a simple logistic regression baseline performs worse when the query and the corresponding span of the context differ a lot in their syntactic structure, but humans do not have that problem. I wanted to see how explicitly incorporating information about parts of speech, named entities and dependency structures influences the model performance: is the model strong enough as it is to infer all necessary information on its own, or will explicit tagging help? In this section, I describe my experiments with different kinds of tagging.

2.3.1 Part-of-Speech Tagging

First, I tried explicitly incorporating part-of-speech tags into the annotation. I used POS-tagging performed by the spaCy[6] library, which is used by AllenNLP for tokenization. The POS-tags were represented in the model using embeddings of size 20, which were concatenated to the word embeddings (therefore, the input dimension of the subsequent model layer also had to be increased).

The model augmented with POS-tags showed validation EM score of **68.1** and F1 score **77.8**. Training progress is shown in Figure 4. Both scores are almost a point higher than those of the non-augmented model. While the change is not drastic, we can conclude that explicit information about parts of speech can improve the model.

[6] https://github.com/explosion/spaCy

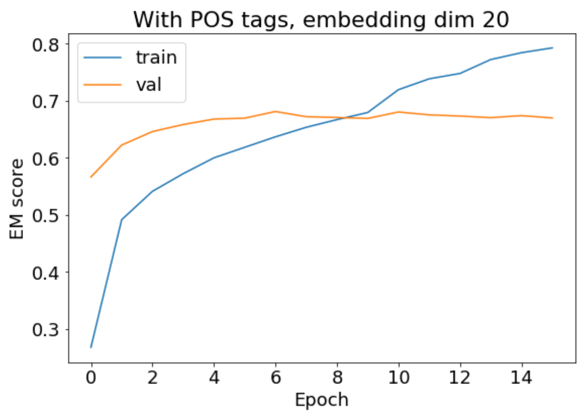

Figure 4: Training progress of the AllenNLP BiDAF model with POS tagging. The figure shows the exact match (EM) score on training and validation sets during training.

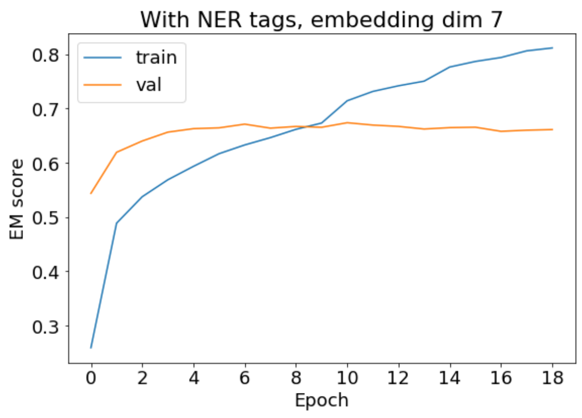

Figure 5: Training progress of the AllenNLP BiDAF model with NER tagging. The figure shows the exact match (EM) score on training and validation sets during training.

2.3.2 Named Entity Recognition

At this stage I tried providing the model with explicit named entity recognition tags. These were also obtained from the spaCy library together with tokenization. Tag embeddings of size 7 were used.

The model yielded best validation EM score of **67.4** and F1 score of **77.2**, which is just a little better than the original retrained model. Learning curves shown in Figure 5.

2.3.3 Dependency Parsing

It is also possible to obtain dependency labels with spaCy. I did that, and represented labels with embeddings of size 20, concatenating them to the word embeddings as well.

This augmentation also improved the model just a little, to EM of **67.6** and F1 of **77.4**. Training progress shown in Figure 6. Inaccuracies of the parser could also influence the result, as dependency parsing is a higher-level and harder task than part of speech tagging or named entity recognition.

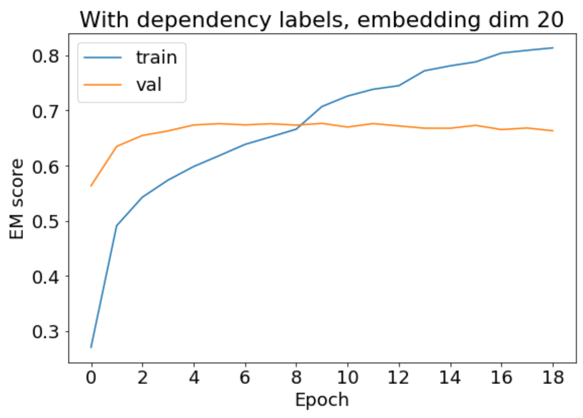

Figure 6: Training progress of the AllenNLP BiDAF model with dependency labels. The figure shows the exact match (EM) score on training and validation sets during training.

2.4 LSTM Character Encoder

The BiDAF model uses convolutional layers as the character-level encoder on top of character embeddings. I would rather expect to see a recurrent architecture, such as LSTM, in the encoder. This modification does not have a solid motivation, but I decided to try replacing the CNN with an LSTM as well and just see how it would influence the performance. I used a 1-layer LSTM with input size 16 (the size of character embeddings) and hidden size 20.

Learning curves are shown in Figure 7. The model with LSTM character encoder shows EM score **66.9** and F1 score **76.4**, which is not very different from the retrained model of the original configuration (around half a point difference on both scores), but still a little worse. It appears that replacing CNN encoder with LSTM does not make a huge difference, and perhaps hyperparameter tuning could lead this model to yield results similar to the original ones.

3 Conclusion

I tried out several modifications of the original model:

- using ELMo embeggings instead of GloVe,

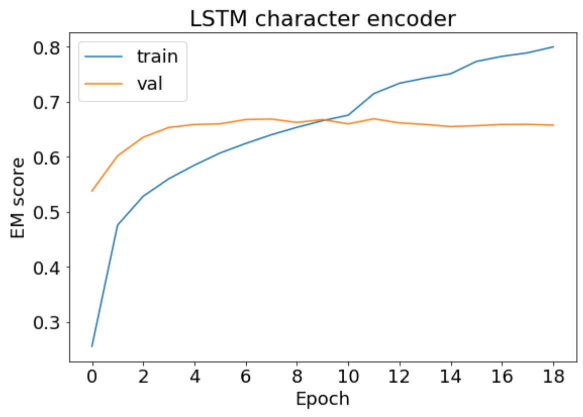

Figure 7: Training progress of the AllenNLP BiDAF model with LSTM character encoder. The figure shows the exact match (EM) score on training and validation sets during training.

- learning word embeddings during model training,
- augmenting the model with part of speech tags,
- named entity recognition information,
- dependency labels,
- using an LSTM character encoder instead of CNN.

The best result was shown by the model with ELMo embeddings. Using trainable word embeddings significantly worsened the performance. This shows that the model underfits when it has to learn appropriate embeddings on its own. With LSTM encoder, the scores decreased a little, which could probably be aided by hyperparameter tuning. All three types of tagging improved the performance a little, best of all part of speech tagging, which means that, while quite strong from the beginning, the model can still benefit from additional explicit linguistic information.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

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