Squadobots and Decepticonvs

Amine Elhafsi
amine@stanford.edu

Patrick Washington
phw@stanford.edu

Problem and Background

- We consider the problem of Question Answering (QA) on the Stanford Question Answering Dataset (SQuAD) 2.0.
- The objective is to design a system that answers a **question** using the provided **context** information.
- Formally, given a question of N words, $[q_1, \ldots, q_N]$, and a context paragraph of M words, $[c_1, \ldots, c_M]$, the QA system should return a span of context words $[c_{\text{start}}, \ldots, c_{\text{end}}]$ as the answer or an empty span if unanswerable.
- The QA problem is relevant to many modern day technologies ranging from digital assistants like Siri and Alexa to the handling of Google search queries.
- This problem involves addressing many open challenges in Natural Language Processing (NLP) such as text comprehension, sequence modeling, and information retrieval.

Methods

- We trained a deep neural network that is adapted from the provided BiDAF model implementation.
- We preserved the BiDAF model structure but **investigated the impact of various design choices** on F1/EM performance metrics including:
  - introducing **character embeddings**,
  - replacing LSTM layers with **Transformer blocks** for improved global context modeling,
  - introducing **convolution layers** for improved local context modeling, and
  - model **pretraining**.
- **Pretraining** was performed using the SQuAD 2.0 dataset.
  - We corrupted the context with random word and character vectors and train the model to reproduce the true context.
  - We used an adaptive softmax layer to output the context without the corruption.
- **Remarks:**
  - Introducing character embeddings produces the largest performance improvement.
  - Transformers perform similarly to LSTMs for the hidden sizes permitted by our hardware memory constraints.
  - Convolution layers **after** the Attention Flow Layer provide small performance improvements.
  - Convolution layers **before** the Attention Flow Layer appear to smear **per-word information**, hurting performance.
  - Pretraining also yields a minor performance improvement.

Best Model

- Our best performing model (Fig. 1) uses word and character embeddings and a convolution layer between the Attention Flow and Modeling Layers.
- We found using Transformers for the Contextual Embed and Modeling Layers performed similarly.
- This model was pretrained for 12 epochs with the corrupted input and fine-tuned on the QA task for 18 epochs.
- Training was performed with a batch size of 64. We use Adadelta as the optimizer with a fixed learning rate of 0.5. Training took approximately 3 hours on an Nvidia GeForce RTX 2080 Ti.

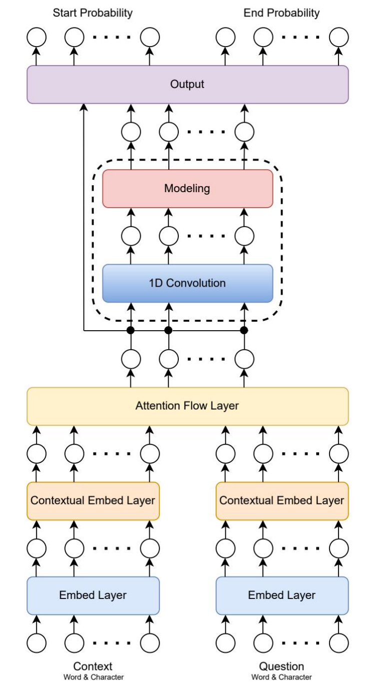
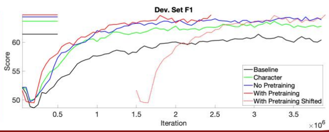


Fig. 1: Model architecture for the best performing model. This model takes both word and character embeddings as inputs and introduces a convolution layer following the Attention Flow Layer. This model was pretrained to reconstruct corrupted context data from SQuAD 2.0.

Experiments

- Performance on the dev set increased quickly for the first 10 epochs and slowed afterward.
- The plot shows that the pretrained model had slightly faster task-specific learning but similar overall time. The dotted line starts at the number of iterations of pretraining to demonstrate the total time.
- The lines on the left edge show the maximum achieved score for each model. As the model improved, the incremental changes got smaller.

Results	AvNA	F1	EM
Baseline Dev	68.46	61.39	57.81
Baseline+Character Dev	70.02	63.63	60.04
Transformer Embedding Dev	69.82	63.77	60.36
Convolution Dev	71.1	64.39	60.66
Our Best Model Dev	**71.48**	**64.97**	**61.55**
Our Best Model Test	-	63.94	60.37



Analysis

- **Character Embedding**
  - **Question**: What is a ligand on the cell surface that is upregulated after helper T cell activation?
  - **Context**: "…helper T cell activation causes an upregulation of molecules expressed on the T cell's surface, such as CD40 ligand…"
  - **Prediction**: CD40 ligand ✅
  - Character-level representation was required to figure out CD40, since it is a rare word.
- **Understanding vs. Word Finding**
  - **Question**: What King and former Huguenot looked out for the welfare of the group?
  - **Context**: "…Henry IV, a Huguenot before converting to Catholicism, who had protected Protestants through the Edict of Nantes."
  - **Prediction**: Henry IV ✅
  - The model understood similarities between "looked out for the welfare" and "protected."
  - It figured out that Henry IV was a King based on other context, despite never using the word King.
- **Trouble with Modifiers such as Ownership**
  - **Question**: What sort of motion did Newcomen's steam engine continuously produce?
  - **Context**: "… James Watt patented a steam engine that produced continuous rotary motion. …"
  - **Prediction**: rotary motion ❌
  - It understood that "rotary motion" is linked to "steam engine" but incorrectly credited Newcomen.
  - In another example where the question asked about Watt, the model gave the correct answer.

Conclusions

- **Subword modeling** is crucial for questions pertaining to specialized terminology, numerical entities, or obscure words.
- Transformers seem to require significantly more parameters than LSTMs to see performance benefits.
- Convolution layers before the Attention Flow Layer appear to **smear information where per-word information** seems important.
- **After Attention Flow, convolutions help** aggregate local context for answers.
- Pretraining yielded a minor performance improvement, but would likely be **more useful with a larger unlabeled data set**.