ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

DISI

Master in Artificial Intelligence

Andrea Pinto - andrea.pinto2@studio.unibo.it

Giorgio Buzzanca - giorgio.buzzanca@studio.unibo.it

Implementing QANet For Question Answering

NLP Project Report

Academic Year 2021 - 2022

Contents

Chapter 1

Executive Summary

We started by analyzing the SQuAD dataset[1], seeking a way to formalize the task of question answering, to get an idea of its properties and the preprocessing steps needed, as well as possible features to feed a model with.

After a shallow literature review on the main methods to tackle this task, we decided to go for an encoder-decoder-like architecture, as LSTMs with attention[2], transformers[3], or BERT[4]-like architectures.

In particular, given the huge training cost and the need to use more datasets associated with transformers, and the lack of parallelizability of recurrent layers, we decided to opt for an advanced architecture easier to train, which adopts a completely different approach.

Indeed, QANet[5] consists of only convolutional layers and relies on context-to-query attention and self-attention[3] After a thorough examination of the architecture, we were able to implement it from scratch, making personal choices where details were lacking in the paper.

We performed a full training/evaluation cycle by splitting the dataset we were provided with, and analyzed the predictions made by the model.

Chapter 2

Background

20202040

Andrea (alla fine)

Chapter 3

System Description

Come Ã¨ fatta l'architettura? Quanti parametri ha? A cosa servono i vari layers?

Andrea

Chapter 4

Experimental Setup And Results

Initially, the dataset was split into a training set, accounting for 95% of the total, and a validation set. We performed the split preserving the logical separation between different topics (i.e. title keywords).

Every analysis was performed on the training set, without ever inspecting the validation set.

We decided to ignore contexts longer than 400 tokens, since those constitute only around 0.07% of the total, and questions longer than 60 tokens, since the longest question in the training set is composed of 40 tokens, but longer questions could be present in the test set.

Then, we tokenized contexts and queries, leaving also the opportunity for the user to compute words' POS tags as additional features. For tokenization and POS tagging the spaCy library is used, with the pipeline optimized for accuracy.

In order to correctly retrieve the answer text at inference time, we also included the offset of each token of the context among the features of the dataset.

Then, we built the embedding matrices for words and characters.

The word embedding matrix was built loading pretrained GloVe[6] embeddings of dimension 300 of only those words appearing at least once in the training set.

Hyperparameter	Value
Number of 2D convolutions applied to character embedding	1
Kernel size of 2D convolutions applied to character embedding	5
Character embedding dimension after 2D convolutions application	200
Number of linear Highway layers	2
Model hidden dimension (i.e. Resized embedding dimension)	128
Number of attention heads in the multi-head self-attention layer	8
Number of 1D convolutions performed inside the embedding encoder layer	4
Kernel size of 1D convolutions performed inside the embedding encoder layer	7
Number of encoder blocks in the embedding encoder layer	1
Number of 1D convolutions performed inside the model encoder layer	2
Kernel size of 1D convolutions performed inside the model encoder layer	5
Number of encoder blocks in the embedding encoder layer	7

Table 4.1: Hyperparameters' values.

The character embedding matrix, of dimension 200, was randomly initialized drawing weights from a uniform distribution with zero mean and standard deviation equal to 0.1.

We reserved the first two index of these matrix for the <PAD>and <UNK>, to which all OOV words and characters were mapped, tokens respectively. The corresponding embeddings of these two tokens were initialized as zero vectors.

Both the mappings of words and characters to indexes and the embedding matrices were stored on disk to be able to load them at anytime.

Word embeddings were kept fixed during training, unlike character embeddings which were trainable.

A summary of the hyperparameters of the network can be found in the table 4.1.

As in [5], we used the ADAM optimizer[7], with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 1e^{-7}$. We used a scheduler for the learning rate, increasing it from 0 to $1e^{-3}$ during the first 1000 batches, where $lr_i = \frac{1}{log(1000)} * log(i+1)$, and then maintaining it constant. We also applied exponential moving average with a decay rate of 0.9999 to all trainable parameters.

The model was trained for 10 epochs on a Google Cloud Platform instance equipped with a GPU Nvidia A100 40GB, with a batch size of 32.

In addition, given that only 2 answers are longer than 30 tokens, we decided to consider only answer spans of length smaller or equal to 30 during evaluation.

Chapter 5

Analysis Of The Results

Giorgio

Discussion

Andrea e Giorgio

Bibliography

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

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

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[5] 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.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. 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, October 2014. Association for Computational Linguistics.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.