Project 2
SQuAD machine comprehension report
CS5242

Group 16

November 16, 2017

Partners: Kyaw Zaw Lin(E0218306)
Luu Tuan Nghia(E0013431)

1 Problem definition

Reading texts and answering questions are common tasks in NLP. In this project, we developed a model that is able to answer questions automatically given an associated context paragraph. With the developed model, each answer is a set of consecutive words from the original context paragraph. The model architecture allows it to work with any language, however, because word vectors are processed using an English dictionary, the model will only work with English language for now.
Based on literature review,the majority of state-of-the-art works follow this general design shown in Fig 1, where questions and context are separately represented by different embeddings and the model output start and end probability.

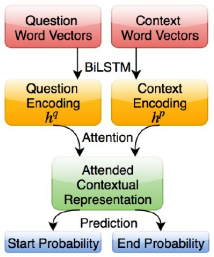

Figure 1: General model for QA tasks

2 Models investigated

The model we developed is the combination of 3 different models: Bidirectional Attention Flow(BiDAF) [1], DrQA[2] and Simple Recurrent Unit(SRU) [3]. All 3 models were trained separately so that they can handle question answering tasks independently. The results of all 3 models were then combined by voting with some weights. The weights were chosen so that the models with higher accuracy when testing alone would have higher weights. The intuition was to trust better models more. The details of individual model architectures are explained below.

2.1 BiDAF

The model is based on LSTM, its architecture is shown in Fig 2. The architecture

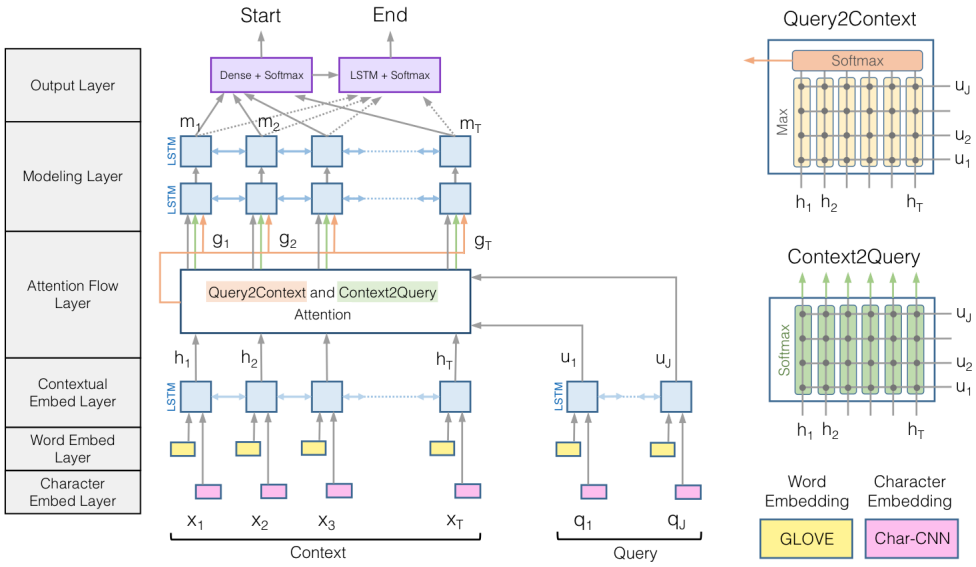

Figure 2: BiDAF Framework

consists of 6 layers:

- Character embedding layer: this layer embeds words into high dimensional vectors using character-level CNNs.
- Word-embedding layer: maps words into high dimensional vectors using GloVe (pre-trained word embedding model).
- Contextual layer: Using LSTMs in both directions to capture temporal relationship between words.
- Attention flow layer: combine context and query vectors and produce a set of query-aware vectors.
- Modelling layer: Using LSTMs to scan the context from the output of attention flow layer.
- Output layer: provides an answer for the query.

2.2 DrQA

DrQA is a system designed to answer open domain questions by using Wikipedia corpus. The system is composed of Document Retriever, which is a classical information retrieval module and the Document Reader, which is the machine comprehension component used

on the rough results of Document Retriever. Document Reader is trained on SQuAD and is generally similar in overall design with bi-att-flow with the following differences:

- Paragraph encoding: Each paragraph is encoded by a multi-layer bidirectional LSTM into vectors which represent the main context of the paragraph. In order to do so, DrQA uses the following features:
  - Embedded words: words are embedded using GloVe into 300-dimensional vectors.
  - Exact match: indicating whether a paragraph matches exactly with a question word.
  - Token features: embed useful information such as term frequency.
  - Aligned question embedding: similar to exact match, it specifies the similarity between a paragraph and a question.
- Question encoding: Embed each question into a vector using a simple LSTM with embed words from the question.

After encoding paragraphs and questions, Document Reader predicts the span of the answer for each question by calculating the probability of start index and end index of the answer.

2.3 Simple Recurrent Unit (SRU)

SRU is a technique to improve the speed of RNNs. Typical RNNs which use LSTMs to compete with gradient vanishing/exploding over a long distance usually take a long time to train. The reason is that each network layer depends on the output of the previous layer. Therefore the layers cannot be calculated in parallel. SRU overcomes this issue by making the gate computation depend only on the current layers input. The results of all layers are combined via a fast recurrent structure. The difference between SRU models and LTSM models is shown in Fig 3. In our project, we used a model of DrQA which made use of SRU technique in hope of speeding up the training. This model performs slightly better than DrQA due to different set of hyperparameters.

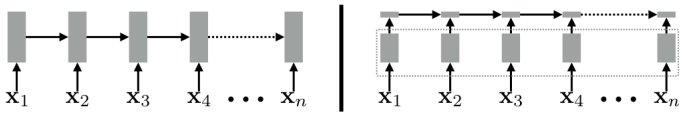

Figure 3: LSTM architecture is shown on the left, and SRU architecture is shown on the right.

3 Ensembling

Our ensembling strategy is a simple weighted voting requiring only the csv file uploaded to kaggle. It can be described as follow:

$$y = \arg\max_{a_i \in A} \sum_{j=1}^{m} w_j X_A(M_j(x) = a_i) \tag{1}$$

where the indicator function $X_A(M_j(x) = a_i)$ equals 1 when $M_j(x) = a_i$ and 0 otherwise. A is the set of all answers and w_j is the weight associated with model j. We have considered using probability directly to perform soft weighting. However, we abandon the idea as we are unsure how to calibrate the probabilities from different models.
The weights are first initialized as accuracy on validation set and then were slightly tweaked by submission against Kaggle test platform. In our final submission, we used 6 models, two from each primary models(BiDAF,DrQA,DrQA+SRU). Interestingly, even though BiDAF performs worst, without it the ensemble fails to obtain the best performance. This is likely to do the fact that DrQA and DrQA+SRU share much in common and produce highly correlated answers. Thus low correlated predictions from BiDAF can increase the final performance.

4 Dataset pre-processing description

The projects used NUS dataset, which was a subset of the Stanford Question Answering Dataset (SQuAD). The dataset consists of 295 context paragraphs, each with a set of questions. In order to use the dataset efficiently, every word has to be embedded into a word vector so that the distance between 2 word vectors can represent the relationship between 2 words. Uncommon words were not considered as there was too little information about them to make a good use of them. Common words are embedded by using GLoVe (Global Vectors for Word Representation) and Character level CNN. Additional features such as pos tags, named entity and term frequency are used in DrQA.
We have additionally split train dataset with 80:20 ratio into train and validation sets to perform hyperparameter tuning and model selection. The final models are retrained on full set once the optimal parameters are finalized.

5 Experiments

The training process was done on 2 machines with 2 TitanX and 2 P40 gpus. Each model was trained separately. It took about 15-20 hours to complete training a model. Different hyperparameters were used during the training process in order to find out the optimum parameters. The parameters which were tried during the training process and their effects are as follow:

- Batch size[larger sizes almost always give better performance]

Table 1: Performance of different models based on public leaderboard

Model	EM
BiDAF	.5440
DrQA	.5848
DrQA+SRU	.5934
Ensemble(6 models)	.6177

- Dropout rate[between .2 and .5 is the best]
- Weight decay[always decrease performance, perhaps due to weights being shared across time, maybe we need to try much smaller weight decays than typical cnn]
- Number of layers[almost no effect]
- Hidden layer size[larger size slightly increase performance ¡.5%]
- Different rnn types(rnn,gru,lstm)[lstm performs best]

Final performance of each model and combined ensemble is given in Table 1.

6 Reflection And Findings

We first tackled this problem by starting with publically available SQuAD leaderboard and reviewing the related literature of top performers. We adopted the stance that given the amount of research and effort devoted to this topic, and the difficulties researchers have faced in reproducing rnet [1], we feel it is unwise to start coding from scratch and instead opted to use openly available implementations. Our approach can be thought of as a literature review + hyperparameter exploration exercise.
The first issue we found the importance of batch size, initially we trained on small gpus due to limited memory. However, we are unable to reach the reported accuracies. After we have procured access to large ram gpus, we restarted training with large batch sizes and improvements of 6-7% can be observed. We believe this is due to the diverse nature of SQuAD dataset and larger batch sizes help smooth out rugged decision boundaries and subsequently improve generalization.
Secondly, we would like to understand the underlying reason for the performance differences between three primary models we have tested. However, due to the large number of differences between the implementation of BiDAF and DrQA, a systematic comparison is not possible. The main differences are:

- Different GloVE vector sizes(100 vs 300)
- Different nlp preprocessing routines(BiDAF uses nltk/standford and DrQA uses spacy)

[1] `https://yerevann.github.io/2017/08/25/challenges-of-reproducing-r-net-neural-network-using-keras/`

- Different models

Post competition, we have performed further experiments by replacing BiDAF original 100d glove vectors with 300d version and observed about 1% increase in performance. Given that the actual modelling of both implementations follow the overall general design, we think that the difference of 3% should be due to the manually engineered features that drqa have added. This shows that despite the power of deep learning models, they are still inadequate to fully represent the full range machine understanding of text and still requires manually adjusted features to help.
We have also noticed the problem that longer answers tend to be wrong. Although we did not have time to further explore this, we did a small experiment by choosing the shortest answer of all the models. Unfortunately, this significantly decreases performance. If we continue to work on this project, investigating this issue can be resolved will be our future work.

References

[1] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *CoRR*, vol. abs/1611.01603, 2016.

[2] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *CoRR*, vol. abs/1704.00051, 2017.

[3] T. Lei and Y. Zhang, "Training rnns as fast as cnns," *CoRR*, vol. abs/1709.02755, 2017.