1st place solution

posted in Quora Question Pairs 8 months ago

204

Hi everyone !

First of all, thanks to Kaggle and Quora for this tough and exciting competition, it has been a pleasure for us to work on it, we learnt a lot of things, thank you !

We also thank our wives/GFs for their patience while we were coding on sunny week-ends :)

We also want to deeply congratulate all competitors, especially Depp Learning team, who really scared us until the last moment !

Among us, Maximilien is a PhD student in a Chair of research (Data Analytics & Models in Insurance) between BNP Paribas Cardif and Lyon University, and the rest of us are colleagues at the Datalab of Cardif. Being all based in Paris surely helped for efficient team work.

1/ Features

We distinguish three kind of features : embedding features, classical text mining features and structural features.
Embedding features

- Word embeddings (Word2Vec)
- Sentence embeddings (Doc2Vec, Sent2Vec)
- Encoded question pair using dense layer from ESIM model trained on SNLI

Remark: Sentence embeddings were challenged but were not that much informative compared to Word2Vec

Classical text mining features

- Similarity measures on LDA and LSI embeddings.
- Similarity measures on bag of character n-grams (TFIDF reweighted or not) from 1 to 8 grams.
- Abhishek's and owl's kindly shared features.
- Edit and sequence matching distances, percentage of common tokens up to 1, 2, ..., 6 when question ends the same, or starts the same
- Length of questions, diff of length
- Number of capital letters, question marks etc...

- Indicators for Question 1/2 starting with "Are", "Can", "How" etc... and all mathematical engineering corresponding

We also used stanford corenlp to tokenizer, postagger and ner to preprocessing text input for some deep learning models.

Structural features (i.e. from graph)

- We built density features from the graph built from the edges between pairs of questions inside train and test datasets concatenated. We had counts of neighbors of question 1, question 2, the min, the max, intersections, unions, shortest path length when main edge cut....
- We went further and built density features to count the neighbors of the questions neighbors... and questions neighbors neighbors .. (inception). We also counted neighbors of higher order which also were neighbors of lower order (loops).
- We tried different graph structures : we built undirected and directed graphs (edges directed from question 1 to question 2), we also tried to separate the density features of question 1 from the features of question 2 to generate non commutative features in addition to commutative ones.
- We built features describing the connex subgraph the pair belonged to : Number of edges, number of nodes, % of edge in train
- We also computed the same features on sub graphs built only from the edges of questions which both appear more than once. What we wanted was to remove fake questions which we thought were damaging the graph features by changing its structure.
- Finally as other teams, we weighted our graphs with some of our initial models. We tried logit and rescaled prediction but raw prediction worked best. We also weighted the graphs with one of our similarity features.

2/ Models

We worked on two main architectures for our NNets : **Siamese** and **Attention** Neural Networks.

- Siamese LSTM with pretrained Glove embedding
- Decomposable attention (https://arxiv.org/abs/1606.01933) with pretrained FastText embedding. This model achieve ~0.3 on cv
- ESIM (https://arxiv.org/abs/1609.06038) with pretrained FastText embedding. This is our best pure Deep Learning NLP model, it achieves ~0.27 on CV. However this model take too long to run, we only add it once in the first stacking layer
- We noticed that DL complex architecture contributed in the first stacking layer but did not do better than simple MLP on second layer

One of the key issue was to select and incorporate some of our traditional features into these networks.
We used FastText and Glove pre-trained embeddings with trainable=False, since our attempts to fine-tune them didn't lead to any improvement.
Eventually, neural networks trained on both text sequences and our graph / text mining features proved to be our best single models.
In the end, we also tried to train siamese models on a character level to provide further diversity to our stacking, but it is hard to tell whether it was really helpful.
We then tried more classical algorithms to exploit graphical features, such as XGB / LGBM which worked pretty well as usual.

3/ Rescaling

To balance with the difference of target distribution between train and test, we also looked a bit closer on the analysis of sweezyjeezy (thanks again for your contribution which helped almost all the participants) posted here :

https://www.kaggle.com/c/quora-question-pairs/discussion/31179

We figured we could reduce the log loss by optimizing the rescale. We did not found a better hypothesis to modelize the distribution of the data in the test dataset, but we made it more accurate by using it on local subsamples of the data.

We found that the train/test biais is very different on 3 perimeters:

- Perimeter 1: qid1_count = qid2_count = 1
- Perimeter 2: min_qid_count = 1 & max_qid_count > 1
- Perimeter 3: min_qid_count > 1

We tried the public rescale and the same rescale but by perimeter. It works well for the first layer models but as we go deeper in our stacking, we found that the public rescale is not strong enough while the rescale by perimeter is too strong. We optimized our rescale so that it falls in the middle between these 2 methods and it helped to gain ~0.001 comparing to public rescale.

4/ Stacking

We made a 4 layers stacking :

- Layer 1 : Around 300 models, Paul and Lam's neural nets, and classical algorithms like XGB, LGBM, which worked pretty well, and a lot of Scikit-learn classification algorithms (ET, RF, KNN, etc.)
- Layer 2 : Around 150 models using:
    - All the inputs features

    - Predictions of aAll the algorithms above
    - We added hidden layers of the best L1 pure text ESIM model
- Layer 3 : 2 Linear models
    - Ridge by perimeter (3 perimeters were created, based on min/max degrees) on 3 least Spearman correlated L2 predictions
    - Lasso with logit preprocessing of all L1 and L2 predictions
- Layer 4 : Blend
    - 55/45, based on public LB score (final and best submission)

Ohad Zadok • (241st in this Competition) • 8 months ago • Options • Reply 3

Nice work! How did you construct the graph? What did you based on for connecting two nodes?

Maximilien... • 8 months ago • Options • Reply 2

We used the Python library networkx (if you are a R user I believe you can do the same stuff with igraph).
The input of graph construction was just the couple of questions in a row (so each row represents an edge, and each question a node). I think you might grab more information on other topics dedicated on this. This will probably be explained more exhaustively ;-)

mubi • (236th in this Competition) • 8 months ago • Options • Reply 2

Thanks for the explanation. A further question, is each row always an edge or only when the two questions are duplicates?--OK, i figure out later, it should be the former considering that train and test datasets are concatenated.

Maximilien... • 8 months ago • Options • Reply 3

@BrunoGuilherme @RVK You right I cannot find anymore what I saw on the discussion, si I'll try to be clear here^^
A graph is an ensemble of vertices and edges, you can find whatever you want here : https://en.wikipedia.org/wiki/Graph_theory
To construct a graph, you need at least 2 columns, i.e. and ensemble [(V1_1,

V2_1), ..., (V1_n, V2_n)]. Here (V1_i, V2_i) represents the i-th edge of the graph connecting V1_i and V2_i.

Note that for i != j, you may have V1_i == V1_j, and so on with V2 etc. And that is what interests us here, because you can draw a lot of properties when two vertices are connected together, and when a vertex is connected to several others.

In this competition, each vertex represents a question, and each edge represents a row in train + test concatenated.

Then you simply explore all what you can do with networkx API, such as collect each vertex's degree, neighbors' degrees and so on.

Massimo Nic... • 8 months ago • Options • Reply 3

Congratulations and thanks for the detailed description! I would be glad if you could answer the following questions.

How many folds did you use for the out of fold predictions?

Did you reuse the same fold indexes between the layers?

Regarding deep models, did you embed POS and NER features and concatenate those vectors to the word representations?

How did you choose the models to keep in level 1 and level 2? Did you check for local CV scores or you just put everything in and let the level 3 models weight the contributions?

Thanks!

Lam Dang • (1st in this Competition) • 8 months ago • Options • Reply 4

We use 10-fold cv, same split between stacking layers.

For deep model, I converted POS and NER into one hot vector. I found that another embedding above that did not help. Also concatenate it to the word embedding did not help, but add it after attention part at comparison layer helped a little on pure NLP model.

For my nnet, since it takes a lot of time (for example at least 30mn per fold for decomposable attention model), I could only afford to take best 10 models from my random search.

For faster ML models we take more to add diversity.

For level 3, we tried both as in the write up.

Jason Miller • (423rd in this Competition) • 8 months ago • Options • Reply 0

Congrats!

What does this mean -- 55/45 final and best submission? Best submission = highest Public LB score. What's the final? The last thing you happened to submit, regardless of its LB score?

Also, could you please explain what you mean by perimeter, or perhaps share a link? I can't find anything about this on Google.

Maximilien... • 8 months ago • Options • Reply 0

Hi Jason,

Our CV/LB score were quite consistent since both scores decreased hand to hand. So when we said 'based on public LB' it also means 'based on our CV score's faith'.

In our 2 L3 models, Lasso performed slightly better than our Ridge, that's why we chose 55% Lasso and 45% Ridge.
And this blend was our final submission (11.27 public LB, 11.5 private LB), which performed better than our two L3 models (11.35 and 11.45 public LB respectively).
I don't know if this answer your first question, feel free to tell me if you need more details.

Perimeters were defined on our section on the rescaling. This corresponds to a partition of train and test, based on min and max degrees (from the graph). And on the train set, we saw that the population didn't behave similarly each other.
That's the reason why we made 3 Ridges, one for each perimeter. What conforted us in our reflexion was the coefficients of the ridge, which were really different for each perimeter, so we judged that it was good to predict each perimeter independantly.

tamar • (25th in this Competition) • 8 months ago • Options • Reply 1

How did you manage to run the 300 layer 1 models and 150 layer 2 models with cross validation and test isn't it taking too long ?

Maximilien... • 8 months ago • Options • Reply 7

The idea was to select some groups of features, especially without our golden features. That would force each model to grab the maximum signal with less important features.
For example, if you let all your magic features, XGB will see it really quickly and will not explore orthogonal signal.

So selecting groups of features (~10% of our total pool of features) reduce the total computation time quite a lot on the one hand, grabbing additional signal (less important, but additional anyway), si that's quite vertuous.

On the other hand, except for KNN who took us aroud 24 hours to predict, all the algorithms we used were quite fast. We didn't use SVM for example.

Same idea for Layer 2.