Selecting Best Answers from Question-Answers Pairs

Anonymous Author(s)
Affiliation
Address
`email`

1 Introduction

Question answering is considered as a huge research area in artificial intelligence. In this work, I am particularly interested not in creating answers to factoid questions, but rather selecting the best answer given a list of answers to a non-factoid question. I will approach this problem in two steps. Firstly I will train the model to learn a good answer pattern independent of the context of the question. The assumption I will use is that for question-answers pairs that contain less than 10 answers, most of the answers are question-related. Then I will add context of the question into the model and compare the evaluation result of two different models.

2 Problem Statement

The dataset I used is a subset of the Yahoo! Answers corpus from a 10/25/2007 dump. It is a small subset of the questions, selected for their linguistic properties (for example they all start with ”how to—do—did—does—can—would—could—should”. Additionally, questions and answers of obvious low quality are removed, i.e., only questions and answers that have at least four words, out of which at least one is a noun and at least one is a verb are remained. The final subset contains 142,627 questions and their answers.

In addition of question and answer text, the corpus contains a small amount of meta data, i.e., which answer was selected as the best answer, and the category and sub-category that was assigned to this question. Any noise that is not human language, such as ”$<$ br $>$” and url are filtered out during the parsing phase.

The evalution is based on the accuracy of predicting the best answer out of an answer list. The second model is expected to outperform the first model. While the exact advantage of the second model is unclear, both models are expected to outperform the baseline, which I will explain in section 4 (Preliminary Experiment).

3 Technical Approach and Models

The method I intended to use is the bag of words model. However I did not expect the bag of words model to have relatively high performance, because it does not capture the length of the question, which I will explain in section 4. Therefore I used the recurrent neural network to replace the bag of words model to see if there could be performance increase.

4 Preliminary Experiment

One important question to answer about the data set is whether this dataset is suitable for a project on answer selection. To answer this question I ploted out the cumulative distribution function of the number of answers. From figure 1, more than 80% of the question-answers pairs contain less than

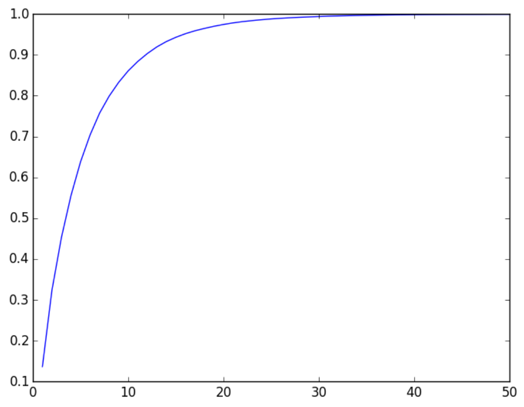

Figure 1: number of answers - percentile

10 answers and over 50% of the question-answers pairs contain less than 5 answers. Roughly 10% of the question-answers pairs contain just one answer, and I will not use those pairs for evaluation.

One interesting feature of a good answer in the real world is the length of the answer. Therefore, instead of using random guess as the baseline, I will use the "picking the longest answer" as the baseline. The baseline is displayed in Figure 2. The baseline accuracy of the model is 55% when all the

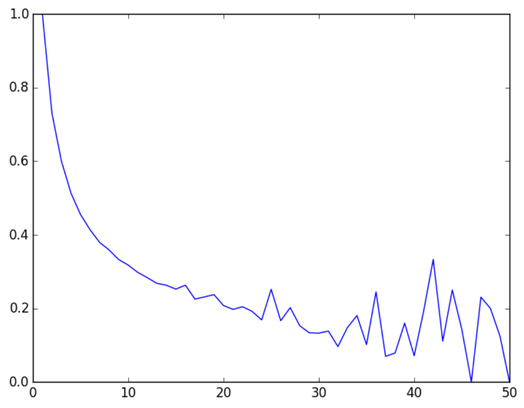

Figure 2: number of answers - accuracy

question-answers pairs are included. The baseline is relatively good considering that it outperforms random guess by maintaining 0.2 accuracy even when the number of answers is 20.

5 Details

5.1 Parsing

I use nltk library to tokenize all the answers and set up the corpus. All the texts within $<$ and $/>$ are filtered out before all the sentences are tokenized. The frequency of each token is then computed, and all the tokens that have frequencies lower than the threshold are set to UNCLEAR. This step is important as it removes lots of tokens with no particular semantic meaning and significantly reduces the size of the corpus. Parsing stage generates a relatively clean corpus with $N = 38921$ tokens.

One category of removed tokens is mathematical formula, because most of the formula appears only once in the dataset. As I am not particularly clear how to handle mathematical formula, I treat those tokens as UNCLEAR, but it is possible to set those tokens to FORMULA in the future research.

5.2 Model Formula

5.3 Model 1

The first model I use is a naive bag of word model. For each answer, the relative frequency of tokens are computed and stored in a vector with of size N. Then I use a two-layer neutral network to predict whether the answer is the best answer.

The output vector $\hat{y}$ is a row vector of length 2. The value at entry 0 represents the probability that such answer is not the best answer.

The error is defined as cross entropy function CE.

$$CE(y, \hat{y}) = -\sum_i y_i log \hat{y}_i$$

The forward propagation is as follows:

$$h = sigmoid(xW_1 + b_1)$$
$$\hat{y} = softmax(hW_2 + b_2)$$

Where x the input vector.

However, because the corpus size is still relatively large, using naive bag of word model takes too long to generate a relatively good result. Each iteration of training with forward propagation plus backward propagation takes more than an hour. Therefore, I switch to model 2 by using sparse matrix computation.

5.4 Model 2

The main model I use is recurrent neutral network. The word vector is of size $N * d$, where d is the size of the vector representation for each word.

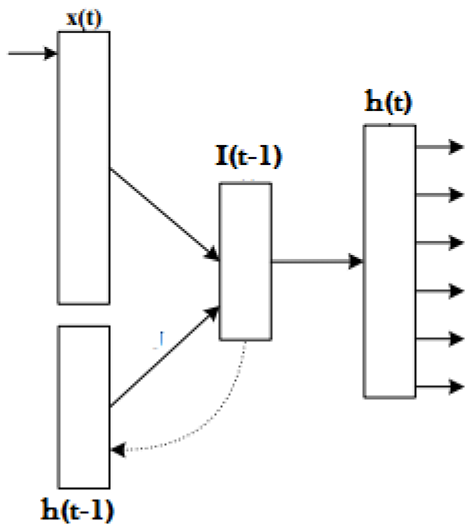

Figure 3: Illustration of the recurrent neutral network

For each answer in the answer set, the model builds up $M - 1$ layers, where M is the total numbr of tokens. For each layer at step t, it takes in the input feature I_{t-1} and generates the output feature h_t. The output feature is of size $d + 1$.

The formula for this model is as follows:

$$I_{t-1} = [tanh(h_{t-1}), x'_t]$$
$$h_t = WI_{t-1}$$
$$x'_t = [x_t, t]$$
$$\hat{y} = \sigma(W^R h_t)$$

Belows are the details associated with this model:

1. x_t is the input word vector at step t.
   -Size d.
2. The adjusted word vector x'_t consists of the word vector for word at step t plus the step t.
   -Size $d + 1$.
3. I_{t-1} is the result of a non-linear layer concatenated with the adjusted word vector.
   -Size $2d + 2$.
4. h_t is the result of a linear layer. h_0 is the word vector at step 0, i.e. $h_0 = x_0$.
   -Size $d + 1$.
5. $\hat{y}$ is the prediction vector. The definition is the same as in model 1: the first entry represents the probability that such answer is not the best answer.
   -Size 2.
6. W: weighted matrix used to condition the input of the current layer. It consists of two submatrices, W_1 and W_2. $W = [W_1, W_2]$
   -Size $2(d + 1)^2$.
7. W_1: weighted matrix used to condition the output of the last layer.
   -Size $(d + 1)^2$.
8. W_2: weighted matrix used to condition the input of the current layer.
   -Size $(d + 1)^2$.
9. σ: the non-linearity function to compute the output probability distribution. In this case $\sigma = softmax$.

The cross entropy error is defined the same as in model 1.

$$CE(y, \hat{y}) = -\sum_i y_i log \hat{y}_i$$

Notice that we do not have prediction vectors associated with those intermediate layers, so the cross entropy error is dependent completely on the prediction vector of the last layer.

The entry of each vector is initialized randomly within range of (0, 0.01).

6 Result and Analysis

Model 2 runs significantly faster than model1, with each iteration of SGD finishing in 1 min because all the updates are sparse update. However, because the dataset is still considerably large, I am unable to completely train the model over the question-answer set. The partial result is still 7% above the base line, and the accuracy is still increasing.

Nonetheless, I would expect Model2 to perform better than the baseline, as the adjusted input vector for each step contains the information of the total number of steps, i.e. the length of the answer. Therefore the model2 is able to potentially use information from all the layers instead of just the input from n closest layers.

One interesting observation from the partial result is that the model generates a lot of false positives. This is partially due to the fact that most of the answers are really similar. For example for the following question-answer pair:

Question : How can I keep my curly hair atleast 6 hours?
– I have straight hair and I like to curl my hair ,(for a change)..I do not want to perm my hair...I use curling iron...but my curls doesn't last very long not more than an hour...is there anyway I can keep my curls longer?

Bestanswer : When you simply curl your hair and don't put anything in it, the curls, unfortuantly, fall out quickly. You should try using hairspray, the Finesse brand isn't expensive and works well. UNCLEAR;Also, some hair types are just very resistant to staying curled. Mine is like that. Thicker, coarser hair seems to be the easiest to get to stay curled.UNCLEAR;It is definitly a good idea to not get a perm, half the time they turn out bad and you are stuck with them. You should just try experimenting with different types of gels and hairsprays and other things like that until you find something that works.UNCLEAR;Good luck!

Answers :

(a) I have the same problem as you. Try towel drying hair, then apply a styling creme. Then try curling your hair. Finish off with hair spray over the curls to set the look.

(b) Try using Hot rollers set with a firm hairspray, Leave them in untill they have gone completely cold. take them out, and shake you curls around gently untill they are placed where you want them, and spary with hair spray. This will give you better hold. Try to aviod touching you hair as much as possible, as this will cause your curls to fall flat faster.

(c) When you simply curl your hair and don't put anything in it, the curls, unfortuantly, fall out quickly. You should try using hairspray, the Finesse brand isn't expensive and works well. UNCLEAR;Also, some hair types are just very resistant to staying curled. Mine is like that. Thicker, coarser hair seems to be the easiest to get to stay curled.UNCLEAR;It is definitly a good idea to not get a perm, half the time they turn out bad and you are stuck with them. You should just try experimenting with different types of gels and hairsprays and other things like that until you find something that works.UNCLEAR;Good luck!!

(d) Get a piece of kitchen roll and folled it half.Then get a piece of hair and rap it round the kitchen roll,put a knot in the kitchen roll.(do it to all the hair that you want to curl.Sleep with tit in and take them out when you wake up. it does work.

The partial result predicts that b and c are both best answer, but in the dataset only answer c is the best answer. This could be explained that this model does not take other answers into account, so rather than picking the answers out of potential answers, it is trying to predict which answer is good, and thereby generates lots of false positives.

7 Conclusion

When the dataset is large, the naive bag of word model is far from ideal because the size of the matrix is too big. Using RNN with sparse matrix multiplication will significantly reduces the number of matrix operations, and is at least 60 times faster when the corpus is large enough.

Whether RNN will pass the baseline is still unclear, because the CE I picked does not really capture the another aspect of this problem, as it is trying to use which answer is the best answer as the heuristics to learn which answer is the good one, and further uses which answer is the good one to predict which one is the best. One potential way to fix it is to use the new predictor i such that:

$$\hat{y}_i[0] = max(\hat{y}_i[0])$$

among all the answers.

For future ideas, the precision of parsing could be improved. For example the line feed $\&\#xa$ is incorrectly parsed as "&", "#", "xa". On the higher level, correctly summarizing the ignored tokens such as mathematical expressions as FORMULA could potentially increase the performance. Integrating RNN model with sentiment analysis (adding sentiment into each word vector) might also be an interesting idea.

Acknowledgments

I would like to thank Yahoo! answer for providing dataset for the question-answer pair analysis.

References

[1] Tomas M., Kai C., Greg C., Jeffrey D. Efficient Estimation of Word Representations in Vector Space
[2] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In International Conference on Machine Learning
ICML, 2008 [3] T. Mikolov, M. Karafiat, L. Burget, J. Cernock y, S. Khudanpur. Recurrent neural network based language model, In: Proceedings of Interspeech, 2010.
[4] A. Mnih, Y.W. Teh. A fast and simple algorithm for training neural probabilistic language models. ICML, 2012.
[5]F. Morin, Y. Bengio: Hierarchical Probabilistic Neural Network Language Model. AISTATS2005.
[6]Mikael Boden. A Guide to Recurrent Neural Networks and Backpropagation. In the Dallas project, 2002.