Predicting Closed Stack Overflow Questions

Levi Franklin
Department of Computer Science
Stanford University
Stanford, CA
lefrankl@stanford.edu

Abstract

Stack Overflow is a website that is used by software developers extensively to ask and answer questions about software development. Each day they reportedly receive over 6000 questions. Due to this high volume of questions it is difficult and time consuming to analyze them all and determine which should be left open and which should be closed for a variety of reasons.

This project aims to provide a method to assist in detecting what questions should be closed and which should remain open. Looking at this problem as a sentence classification problem one can use deep learning and neural network models to train a system that will identify good candidates for closing. With the help of a large dataset provided by Kaggle and Stack Overflow, one can use algorithms that are effective in tasks such as sentiment classification and see how they apply to this task.

To evaluate the success of these models, they are compared to several benchmarks as well as to other competitors in the competition. The multiclass log loss is used as it was in the competition to facilitate this comparison. Different models and sets of word vectors are used and their log losses are compared to determine which system works best.

In addition to the natural language processing that is done on the text of the question, there are some numeric features which are utilized such as the question asker's reputation. Similar tests will be done to see if incorporating these features in the model provides better results. Overall this project aims to show that a good estimate of what questions should be closed can be given by a neural network based prediction model.

1 Introduction

Stack Overflow is a website that is used by developers throughout the world to ask and answer questions that they may have related to software development. Among the reportedly 6000 questions Stack Overflow receives every weekday, many are inadequate to make it onto the site. These questions can be closed for being off topic, not constructive, not a real question, or too localized. With this many questions it is difficult to inspect each one so an automated process to assist in determining what should be closed is very helpful.

The was recently a Kaggle competition to solve this very issue which includes a dataset of questions, metadata about each question, and whether it was left open, or the reason it was closed. This dataset provides a good set of training and testing data to build a neural network to try to create an automated system for making the determination on if a question should be closed.

The first task to consider is how to model the actual words that make up each of the questions in our neural networks. Word vectors are used to represent the words within each question and consist of numeric vectors of some dimension (often 50-100). These word vectors are a representation of how words appear in relation to each other in a large corpus of documents. This project attempts using a pretrained set of word vectors that is well regarded in the field, the GloVe vectors. Additionally, a new set of vectors is trained using the Stack Overflow dataset which incorporates the technical stack overflow terms that may not be in the GloVe set. These two sets of word vectors are compared to see which has better performance.

There are many neural network models that could be applied to this problem. Primarily for this project there are two main neural network models that are considered. The first is a system that averages these word vectors and uses those as the input to a traditional multi-level neural network is used. This has some shortcomings such as not taking into account sentence structure. Secondly a convolutional neural network model is used that passes over the words in each question to consider them in groups. Each of these filters, or convolutions, outputs some estimate that is then pooled together. The result of this is used to make a prediction on what class that input belongs to. This does a better job of including sentence structure but is more complicated and time consuming to run.

The other portion to consider is how to incorporate numeric features in addition to these word vectors. Each question includes some numeric features such as the user's reputation at the time of asking or how many open questions they have at the time of asking. The main method of including these features is taking them on to the end of each word vector so that they can be included in the training. Training with these features is compared to training without to see which is more helpful.

In order to gauge the performance of these models the log loss metric is used. This is the metric that was used during the Kaggle competition which will allow comparison with other competitors. Additionally, the loss of these models will be compared to some baselines. This will include making a uniform prediction of each class as the most naïve baseline, as well as one based purely on the class distribution in our training data. Lastly Stack Overflow has provided a machine learning model they created to compare against other submissions. Overall this project aims to beat both more rudimentary baselines and be near many of the other Kaggle submissions in performance.

2 Background/Related Work

This problem can be treated primarily as a classification problem where the main goal is to take in one stack overflow question and the metadata associated with it, and classify it as one of 5 categories: open, off topic, not constructive, not a real question, or too localized. This problem is a familiar topic in natural language processing called sentence classification. It has been studied before and many people have proposed solutions. A common task that is used as an example in these studies is doing sentiment analysis of things like movie reviews [1]-[4].

The first aspect in these sort of problems is representing the words in these sentences as word vectors. These word vectors are generally generated by creating a matrix of what words are near each other in a large corpus. Usually this corpus consists of things like Wikipedia. One popular word vector representation is the GloVe system which creates a cooccurence matrix and then reduces it using techniques like SVD [5]-[6]. Another model is called word2vec or the skipgram model and learns word vectors on the fly using a predictive model [7]. One can use vectors trained in either way for a neural network model.

Once a method or representing the questions has been determined, it is prudent to investigate what models will best make the associated classification. One easier yet surprisingly powerful solution is to just average the word vectors that are used in the sentences. This does not hold the structure of the sentence but can capture some meaning. Another technique makes use of recursive neural networks which feeds the sentences into a set of similar network levels that share a single weight matrix and feed into the next one, capturing the result of all the words before it [8]. This model suffers from a vanishing gradient issue where it forgets about older words as time goes on and the gradient becomes weaker.

Two other techniques help lessen this gradient problem by using as building blocks of their network a layer that has some “memory”. Gated Feedback Recurrent (GRUs [9]-[10]) and Long Short-Term Memory (LSTMs [11]) neural networks keep track of more information from previous parts of the sentence.

Another system that takes into account structure is convolutional neural networks [3]. These networks take a filter and do a convolution over the sentence. It looks at some number of words and passes along the sentence making predictions based on those words. This can happen multiple times and in the end the results are pooled to make a final prediction. Overall these technique provide several different ways that sentences can be processed and classified based on a set of training data.

3 Approach

The approach that is taken in this paper explores two different models and compares them with various input types. Additionally, it makes use of two different sets of word vectors and compares them to gauge which is more applicable. These different techniques are trained against a large dataset of questions provided by Stack Overflow and are then tested on a separate subset of those questions. The baselines are also computed on that training set and are used to compare these values to the other Kaggle competitors.

The two kinds of word vectors that are considered are the GloVe vectors that are mentioned above and ones that are trained specifically on the corpus of training data that we are using. The GloVe vectors are pre trained and are known to do a good job at expressing the ways that word correlate with one another. However, they are trained on a corpus of Wikipedia and Gigaword [12] so they may be missing some technical terminology that Stack Overflow uses. In an effort to overcome this a large set of Stack Overflow questions is also used to generate another set of word vectors that may be better suited to this task.

Next the first model that is tested should be discussed. This model is an averaging of all the word vectors in each sentence. This resulting word vector is then fed into a neural network and is trained against the dataset of questions. This neural network takes in the average word vector, runs it through some number of hidden layers which multiply it by various weights which are learned during training. The final layer combines all of these into a final prediction of open or closed.

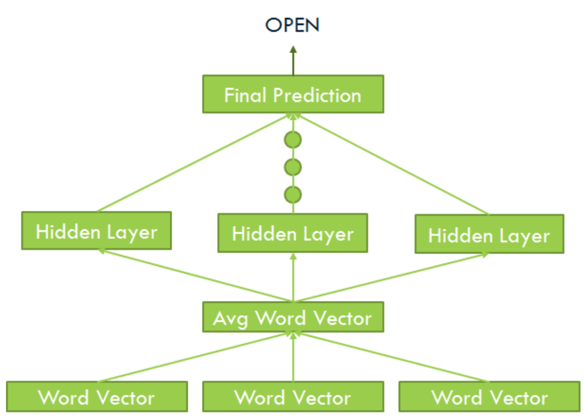

Figure 1. Averaging Word Vectors

The next model that is used is a convolution neural network model. This type of model takes filters of various sizes and applies them over the word vectors. These filters are passed over the words in the sentences in sets that are the size of each filter. Then these filters outputs are pooled together and used to make a prediction much like in the model before. In this model, the actual structure of the sentences matters and influences the final results.

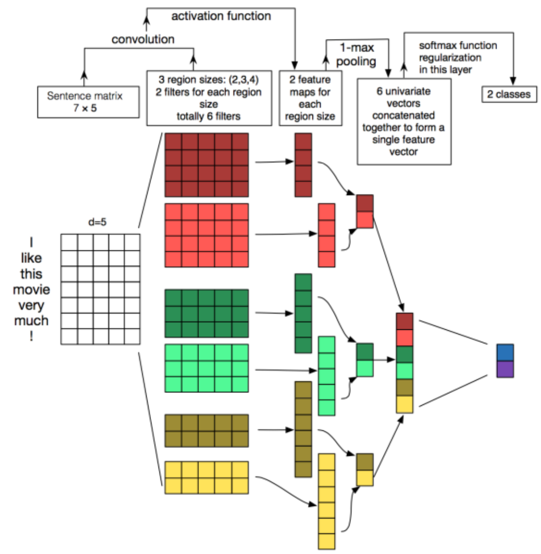

Figure 2. Convolutional Neural Network [13]

The other main consideration is how to incorporate the numeric features in each of these models. As was mentioned before, each question also includes the reputation of the user and the number of other open questions they have. This information can be very helpful in determining if a question should be open as someone with high reputation is much more likely to submit a question correctly. The technique that is used here is to include those two values in the end of the word vectors. For averaging word vectors it is done after they are averaged and for the convolutional network they are added to every vector. This allows them to be included in the neural computations just like any other feature.

Lastly some discussion of the metric that is used to evaluate these models is important. The Kaggle competition used the mutli class log loss. As will be seen later on the dataset that is used has a highly biased distribution towards the open category. This means most all models will achieve high accuracy as well as very similar precision and recall so these metrics are not overly helpful. In order to compare these models to the baselines and the other competitors the log loss is used here as well. The equation for the log loss multiplies the log of the probability for the correct class of each input and averages them all.

$$\text{Log Loss: } -\frac{1}{N}\sum_i y_i \log(\overline{y_i})$$

4 Experiment

4.1 Dataset

The dataset that was used is a very large collection of Stack Overflow questions coming from Kaggle and Stack Overflow. In total it has around 3.4 million questions. The dataset distribution is heavily weighted towards the open class. Stack Overflow has stated that around 6% of their questions are close with even less being marked as closed in the training data. This data was split up as 60% training, 20% dev, and 20% final testing.

Training	2,022,318	60%
Development	674,106	20%
Testing	674,106	20%

Figure 3. Distribution of 3,370,530 questions in dataset

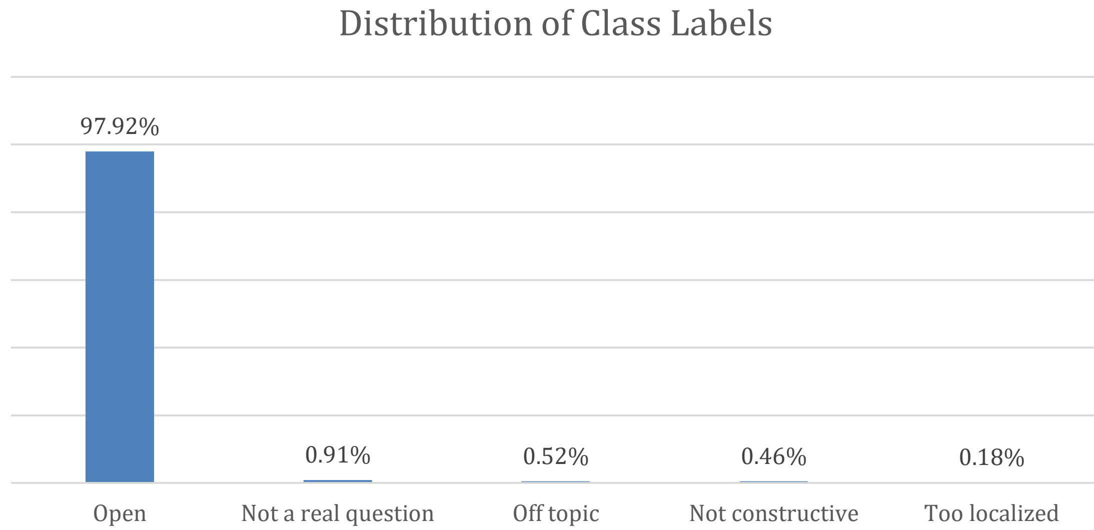

Figure 4. Distribution of class labels

The dataset includes the following features that are useful in training a neural network:

- Question Title
- Question Text
- Submitter's stack overflow reputation
- Number of questions submitter has open at time of submission

Here is a sample entry for a question that was marked as closed:

Users reputation: 32

Users other open question count: 0

Title: List of all .txt file

Text: I want to write a program that give a path in my system and goes to that path and search in that path and sub-directory of path and list all of .txt file. please help me thanks .

Label: not a real question

4.2 Results

To begin the values of log loss on the test set for each of the baselines was collected. As was mentioned before the uniform baselines predicts evenly across all classes and provides a very bottom baseline. Next is a baseline that predicts the same distribution as the dataset distribution that was mentioned above. This is a better baseline in that it predicts open heavily and is almost always right. The goal is to improve upon this baseline. The last baseline is the one based on the model that was given by the Stack Overflow team as their baseline for predicting. This improves upon the frequency distribution baseline substantially and is a good goal for these models to achieve.

The first experiment was to train the averaging model with both GloVe word vectors and the Stack Overflow trained vectors, then compare the final log loss with the baselines and the other Kaggle competitors. Unfortunately, the correct labels for the final submission in the Kaggle competition were not made available so this comparison can only be made in comparison to these three baselines. Secondly, the same experiment is done using the convolutional model. On this experiment however, the GloVe vectors were excluded as it was seen that the Stack Overflow trained word vectors performed better. There was also some work done to try resampling the dataset to compensate for the uneven class distribution, and then readjusting the predictions based on their correct distribution in Stack

Overflow questions. However, this did not prove to help and was abandoned.

Method	**Log Loss**
Uniform Prediction Baseline	1.609
Averaging Stack Overflow Trained Word Vectors	**0.205**
Averaging GloVE Word Vectors + Numeric Features	**0.200**
Convolution Stack Overflow Trained Word Vectors + Numeric Features	**0.212**
Convolution Stack Overflow Trained Word Vectors	**0.190**
Frequency Based Prediction Baseline	0.173
Averaging Stack Overflow Trained Word Vectors + Numeric Features	**0.152**
Stack Overflow Model Prediction Baseline	0.094

Additionally, here is the confusion matrix for the Averaging Stack Overflow Trained Word Vectors + Numeric Features results:

	Not a real question	Not constructive	Off topic	Open	Too Localized
Not a real question	4	8	20	9856	0
Not constructive	3	63	21	3438	0
Off topic	2	10	97	4498	0
Open	6	54	172	653735	1
Too Localized	0	1	0	2114	1

And the precision and recall for each entry:

Label	**Precision**	**Recall**
not a real question	0.2667	0.0004
not constructive	0.4632	0.0179
off topic	0.3129	0.0211
open	0.9705	0.9996
too localized	0.5000	0.0005

Here is an example of correctly classified document as “not a question”:

Title: MySQL Table of UK Area Codes and Names

Text: I need database of Uk cities with their post codes. Any help is appreciated. Thanks for your time..

4 Conclusions

Overall this has been a fairly successful experiment. Though many of the models did not achieve outstanding results, the best performing one did manage to beat both of the starting baselines. This shows that a neural network can indeed be used to help facilitate predicting what questions on

Stack Overflow should be closed. As can be seen by the confusion matrix, it was able to correctly predict questions in every category.

Using this information this models performance can also be compared to those of the other Kaggle competitors. Though it cannot be compared directly due to not having the labels on the test set they used, they can be roughly compared due to the benchmarks. Based on those, the best model here would come around 110th out of 160 submitters. Assuming Kaggle competitors are usually well versed in machine learning, this seems to be a pretty good result.

Interesting questions are what caused the other models to perform not as well. This could be due to bugs in the models or not achieving the best hyperparameters. Each of them have various things that can be tweaked such as number of layers, number of filters, and many others. Though lots of various parameters were used to find the best one, it is possible the best parameters were not achieved. Other future work could entail using other features from stack overflow such as the number of questions the user has had cancelled before. Additionally, the other models that were mentioned during the related work could provide better results.

References

[1] Andrew L. Maas and Andrew Y. Ng (2010) **A Probabilistic Model for Semantic Word Vectors**

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts (2011) **Learning Word Vectors for Sentiment Analysis**.

[3] Yoon Kim (2014) **Convolutional Neural Networks for Sentence Classification**

[4] Quoc Le, Tomas Mikolov (2014) **Distributed Representations of Sentences and Documents**

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. (2014). **GloVe: Global Vectors for Word Representation.**

[6] Eric H. Huang, Richard Socher∗ , Christopher D. Manning, Andrew Y. Ng (2015) **Improving Word Representations via Global Context and Multiple Word Prototypes**

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean (2013) **Distributed Representations of Words and Phrases and their Compositionality**

[8] Toma´s Mikolov, Martin Karafiat, Luka´s Burget, Jan "Honza" Cernock, Sanjeev Khudanpur (2010) **Recurrent neural network based language model**

[9] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio (2014) **Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling**

[10] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio (2015) **Gated Feedback Recurrent Neural Networks**

[11] Sepp Hochreiter, Jurgen Schmidhuber (1999) **Long Short-Term Memory**

[12] Parker, Robert, et al. English Gigaword Fifth Edition LDC2011T07. DVD. **Philadelphia: Linguistic Data Consortium**, 2011.

[13] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ (2015) **UNDERSTANDING CONVOLUTIONAL NEURAL NETWORKS FOR NLP**