Transfer Learning Capstone Proposal - Transfer Learning on Stack Exchange

Kyle LANGE

February 28, 2017

1 Motivation / Background Information

With the vast amount of knowledge and information that is available on the internet, there is a need to accurately classify information according to its most relevant topic so that it can be found in search queries, recommended to users interested in that topic, etc. This problem is also found on question / answer websites such as Stack Exchange, where one would like to be able to tag each query with a set of relevant labels such that the query can be directed to the appropriate person to get an answer and can be found by other users looking for information regarding the same topic. One could also see where an algorithm which can assign labels might be useful for say, labeling bug tickets for a software company so that they can be automatically assigned to the appropriate team. All of this work can be done manually, but it is tedious and expensive. There is therefore a need to develop machine learning algorithms which can automate much of the process.

These types of problems are of personal interest to me, because in my position as a data scientist where I am working on a text classification problem. For my particular problem, I am trying to match a user query with a US Post Office address. Although the address classification problem is much different than a website / query classification problem, many of the same tools such as term frequency / inverse document frequency, cosine similarity scores, etc. can be used.

2 Problem Description

While a supervised learning approach (with a training set of tags and queries) would likely be one way to approach text classification problems, a different approach would be that of transfer learning, where we train models on queries with a completely different topic, and apply knowledge from training these unrelated models to build a model for the queries of interest. In the case of the Kaggle Transfer Learning on Stack Exchange competition, we are given training sets composed of real Stack Exchange Queries with the Title, Text, and Tags from six different topics and asked to predict the tags for a new topic.

3 Data Description

The training data set is composed of stack exchange queries under the the topics of Travel, Cooking, Cryptography, DIY (Do-it-yourself), Biology, and Robotics. The test data set is composed of Physics Queries with Title and Text. The goal is to predict the tags for each physics query. No missing data or additional fields were observed in the text. Some summary statistics for each of the data sets is shown in Table **??**.

4 Evaluation Metric

I plan to use the same evaluation metric as is used in the competition: the mean F1 score. The F1 score is a harmonic average of precision and recall. Algorithms with high precision will produce very few false negatives, but potentially a large number of false positives. Conversely algorithms with high recall will produce very few false positives, but

Table 1: Statistics on Training and Test Data

	Biology	Cooking	Cryptography	DIY	Robotics	Travel	Physics
Number of Queries	52784	61616	41728	103672	11084	77116	245778
Average number of words / query	94	88	142	119	167	96	124
Max number of words in query	2327	2279	7452	1554	5670	1180	3721
Min number of words in query	5	5	3	6	8	4	4
Average number of tags / query	2.5	2.3	2.4	2.3	2.4	3.4	-

potentially a large number of false negatives. The F1 score penalizes very low values for either precision or recall, providing a good balance between false positive rate and false negative rate.

The F1 score for a given query is mathematically described as

$$F1 = \frac{2pr}{p+r}, \tag{1}$$

where p is the precision and r is the recall. These are defined as:

$$p = \frac{tp}{tp+fp} \tag{2}$$

$$r = \frac{tp}{tp+fn} \tag{3}$$

where tp is the number of correctly classified tags, fp is the number of predicted tags which are incorrect, and fn is the number of true tags which are not predicted.

The mean F1 score takes the mean of the F1 scores over each of the queries.

5 Baseline Model

The baseline model is to predict "gravity" as the tag for each physics query, which results in a mean F1 score of 0.01488. This seems like a "majority" class classifier, where each query is tagged with the most common tag; however, in this case, we don't know for certain that this is the majority class classifier since we don't know the tags for the physics queries.

6 Proposal

I see this as a two step problem. The first challenge is to identify the potential tags for the physics queries, as this information is not known a priori. The second challenge is, given the list of potential tags for physics queries, assign the tags to the relevant queries. Using the Pipeline mechanism in sklearn, we can combine the cross-validation and for both the tag identification and query classification, using the mean F1 score for the physics tags as the evaluation metric.

6.1 Identifying Tags

In order to identify the potential tags for physics queries, we will consider the relationship between the term frequency and document frequencies for queries in other fields. In this case we will compute the tf-idf score for each word contained in a set of queries for a given topic. The tf score for term k for topic a will be computed as:

$$tf = \frac{\text{number of times term } k \text{ is used in queries from } a}{\text{number of queries for topic } a} \tag{4}$$

The idf score for term k will be computed as

$$idf = \log\left[\frac{\text{sum of weighted number of all queries}}{\text{weighted number of documents containing word } k}\right] \tag{5}$$

The weighting for the idf score is to account for the fact that each topic has a different number of queries. During the weighting, a factor α_a will be computed for each topic a such that the product $\alpha_a N_a$ (where N_a is the total number of queries for topic a. This α_a factor will then be applied to weight the number of documents containing word k. This can be re-written as:

$$idf_k = \log \left[\frac{\sum_{i=1}^{n} \alpha_i N_i}{\sum_{i=1}^{n} \alpha_i n_{k,i}} \right] \tag{6}$$

In order to identify tags, a threshold tf-idf score will need to be determined. This will be done using cross-validation, where each of the topics with labels will be used separately as a test set, with the remaining topics being used as training data. The average optimum threshold will be used when finding the tags from the physics queries. For two-word tags, a threshold will need to be defined such that each of the individual words has a high enough tf-idf score, and that there is a large enough frequency of the two words appearing together. These will also be computed in the cross-validation process.

7 Query Classification

Once the set of tags has been computed, a first try for the model will be to assign tags to queries where the tag is contained in the query text. However, this may miss queries which have words which are related to but are not the exact text of the tag.

Thus, the following algorithm is proposed. First, each query is examined to see if it has multiple occurrences of one of the tags. For queries with more than one occurrence of the tag, we assign the tag to the query. Once we have finished assigning tags to queries, we consider this set of labeled queries to be our training data. We can then compute tf-idf scores for labeled training data, compute tf-idf scores for each query in the test data, and assign the appropriate label when the cosine similarity score is above a certain threshold.

This threshold will be determined using cross-validation, in the same manner as above. One topic will be used as the test set with the remaining topics being used as the training set and this will be done for each separate topic. Note that the cross-validation will need to be done for each individual tag to classify the query as having the tag or not having the tag.

8 Other Considerations

Throughout the course of this project, there may be other important factors which need to be considered. Some which come to mind include

- It might be better for words in the query question be weighted more highly than words in the query text.
- There may be a better approach for creating two-word tags than just making sure that the individual words have a high enough tf-idf score.
- Different forms of the same word may need to be considered (e.g. frequent and frequency) in order to get good scores.

As the project proceeds, the quality of the results will determine how many of these factors will be incorporated into the model.