Twitter Topic Recommendation with a Deep Learning Model

Names

***Abstract*—???...**

I. Introduction

TWITTER is an online micro-blogging social media platform where users post messages of at most 140 characters in length. Users have developed a convention whereby the topic to which a tweet relates is encoded in the text of the message as a word beginning with a hash symbol. However, only 14.6% of tweets contain such a hashtag.

In this project we attempt to reproduce the modeling approach of Li etal [1], by training a deep neural network on a set of tweets containing hashtags in order to generate a hashtag for tweets that do not contain one. Classifying tweets in this way would potentially enable user interest modeling, recommendation services, and online advertising.

II. Independent Variables

The independent variables in the model are the words and sentences of which a tweet is composed. Approaches such as TF-IDF, kNN, and Naive-Bayes would learn a model using these variables directly. However, there are bottlenecks to such models because they do not make use of the semantic information inherent in words and their composition into sentences. We employ the word2vec technique introduced by Mikolov et al [2], 2013 to capture this meaning.

III. Literature Review

Much recent work in natural language processing (NLP) involves embedding words as vectors using neural networks ([3] Mikolov et al., 2013) and performs classification on top of that ([4] Collobert et al., 2011). We use the conclusions of such work to inform how we build an in-house classifier for the topic of the tweets we collect.

IV. Proposed Approach

We follow the approach of Li etal [1], by first representing each word in the tweet as a numeric vector of multiple dimensions. The vectors are generated from an independent corpus of words using a tool, such as gensim, and then the tweet is converted to sets of these vectors, with one set per sentence. The technique, called word2vec, was originated by Mikolov etal [2] and has the property that similar words appear close to each other in the embedded space and thus capture some of the semantics of the word. A convolutional neural network (CNN) is trained with the word vectors of each sentence of a tweet as the input layer. This yields a set of sentence representations that are fed into an LSTM-RNN to yield a layer representing the tweet which is then fed into a softmax layer whose output is the probability of one of K different hashtags used as class labels.

V. Data Preparation

A. Data Collection

The dataset was collected from Twitter using R "searchTwitter" function. Five topics are selected:*basketball, hockey, baseball, volleyball, tennis* and hashtag is added during collecting, e.g. *#basketball*. The reason why these sports topics is chosen is because they are major sports so that the number of data collected is sufficiently large for the purpose of the project. For instance, golf is one of the sports that were originally selected, but there were too few data points and thus it is removed from the topics. Football also used to be one of the candidates, due to the ambiguity of football and soccer in different region it is also removed. Only five classes are collected since the more classes the slower training would be. Table I below is the final number of data collected,

Topic(#)	basketball	hockey	baseball	tennis	volleyball
Number of tweets	10,000	10,000	10,000	10,000	4,755

TABLE I: Collected Tweets

B. Data Pre-processing

Most of the conventional text pre-processing methods are used, two major differences in this project are that all hashtags of the five sports are removed, since they are the labels for each of the tweets which is exactly the label that the model tries to predict and otherwise the model would learn nothing but appear to be performing really well (high accuracy by just looking at the hashtags); The user names are not removed as they could give clues about which sport topic the tweet belongs to, e.g. "@KingJames" is the username of the basketball star player Lebron James and tweets involving this username are most likely about basketball. So the major pre-processing steps are as follows:

1) Replace *http* links with whitespace.
2) Replace hashtags {#basketball, #hockey, #baseball, #volleyball, #tennis} with whitespace.
3) Replace newline characters such as \r, \n and \r\n with whitespace.
4) Replace special symbols such as #, @, %, &, $, digits and punctuations with whitespace.
5) Remove 'RT's (retweets).
6) Combine multiple whitespaces to one.

7) Remove duplicated tweets.

The order of 2), 3) and 4) matters since if 4) is executed first then 2) and 3) would not work correctly. 1), 5) and 7) are motivated by the same reason as they do not contribute and are noise to the topic classification.

VI. Experiments with Descriptive Analytics

A. Benchmark Experiments

To compare the performance of our model to existing techniques, we ran several experiments to arrive at a benchmark based on these other methods.

In our first pass, we did not remove hashtags for the sports of interest from the tweets in the data pre-processing and cleaning step. We generated a document-term matrix, using the *RTextTools* R library, where each tweet was considered as a document. Frequently occurring terms were chosen as the features to be used for a classifier. To arrive at a reasonable number of features we varied a parameter for removing sparse terms when creating the matrix. With hashtags present, a Naive-Bayes classifier achieved 91% accuracy as per Table II.

Predicted	baseball	basketball	hockey	tennis	volleyball
baseball	2765	5	1	5	142
basketball	4	2679	5	7	246
hockey	5	6	2718	0	353
tennis	8	17	0	2602	398
volleyball	9	20	1	4	1427

TABLE II: Predicted sports vs actual with hashtags kept and *removeSparseTerms* = 0.9

Having realized that the hashtags are essentially the class labels, we understood that their presence in the tweet text causes the classifier to simply recognize the hashtag when training. This results in inflated accuracy and does not reflect the fact that our attempt is to classify tweets that may not include the hashtag word directly. So, in our next pass, we cleaned the tweet text of hashtags, created the document-term matrix to yield more terms, and reapplied the Naive-Bayes classifier. The new accuracy was much lower at only 39% as per Table III.

Predicted	baseball	basketball	hockey	tennis	volleyball
baseball	1005	79	42	19	1773
basketball	89	1494	179	52	1127
hockey	50	60	504	13	2455
tennis	34	84	77	830	2000
volleyball	16	17	12	4	1412

TABLE III: Predicted sports vs actual with hashtags removed and *removeSparseTerms* = 0.985

Since the new accuracy was much lower, we sought to verify the assumptions of the Naive-Bayes model. As the order of words in a tweet is important to understanding its content, the independence assumption may not hold. Also the features are so sparse in the columns of the document-term matrix that the assumption of a Gaussian distribution does not hold. Figure 1 shows the distribution of the term nba from the previous experiment.

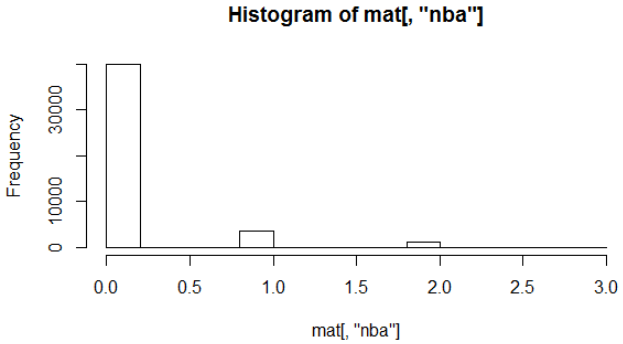

Fig. 1: Non-Gaussian distribution of NBA in document-term matrix

To explore whether we could achieve higher accuracy with other classification techniques, we reduced the data set to 500 tweets per class for a total of 2500, using 70% for training and 30% for test. This reduction enabled our experiments to complete in a reasonable amount of time. For Naive-Bayes we noted a revised accuracy of 38%. Random Forest achieved an accuracy of 27% as per Table 4 and SVM achieved an accuracy of 24% as per Table V .

d_pred	baseball	basketball	hockey	tennis	volleyball
baseball	12	2	14	6	7
basketball	51	52	52	35	54
hockey	16	12	11	5	22
tennis	43	27	47	65	20
volleyball	40	33	46	18	60

TABLE IV: Acutal vs predicted sports as classified by a Random Forest model trained on a reduced dataset

d_pred	baseball	basketball	hockey	tennis	volleyball
baseball	12	5	18	15	11
basketball	66	56	65	43	60
hockey	23	13	12	4	21
tennis	33	24	37	49	18
volleyball	28	28	38	18	53

TABLE V: Actual vs predicted sports as classified by SVM with a radial kernel and trained on a reduced dataset

The SVM classification accuracy for various kernels is plotted in Figure 2.

Clearly, our experiments show that after cleaning tweets of the hashtag labels for the sports under consideration, the best test accuracy attainable with conventional methods is 39%. We now describe our LSTM model and demonstrate its superiority on this task.

VII. Predictive Modeling

A. Feature Engineering/extraction

Each data point in the raw input data set is a tweet that consists a list of words, we convert them to vector representation and use that as the input features to our model. As we have seen previously that using term frequency as the features of

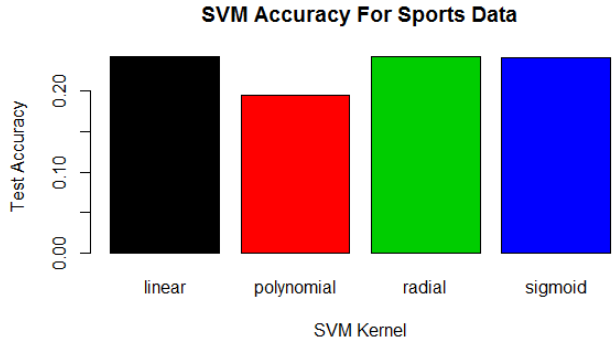

Fig. 2: SVM Test Accuracy as achieved by various kernels on the reduced dataset

each tweet has its limitations since it does not capture semantic information very well. Word2vec techniques, more specifically the Skip-Gram model is used here to convert string of words to vector representation of features Figure 3.

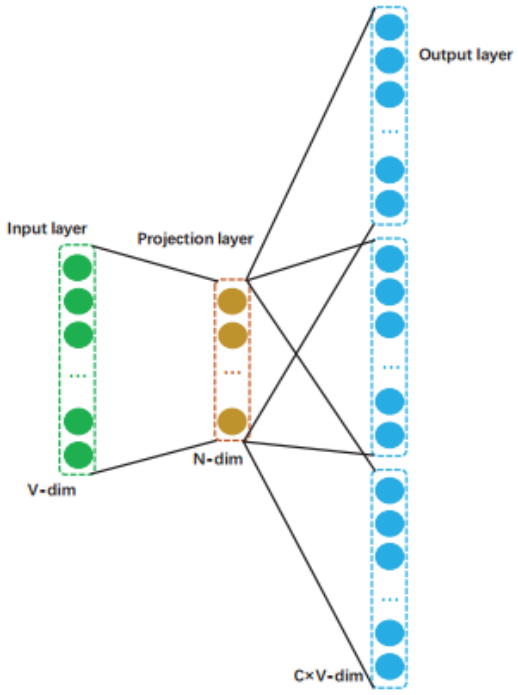

Fig. 3: The skip-gram model architecture [1].

Skip-gram models utilizes a shallow one-layer neural network to learn the representation of words such that the likelihood of obtaining surrendering words from a given words is maximized. In this project, we mapped each unique words into a 600-d low dimensional(comparing to one-hot encoding) vector space, i.e. the number of features used in our project are 600. We could convert the data to even higher dimesion feature spaces, but that would slow the training and not necessary as the model is already performing very well.

B. Model Design

In Li et al [1], they first started with one-hot word representation, pass it through word2vec layers and form distributed word representation. Then a convolutional neural network (CNN) is applied to each sentence in a tweet to extract local semantic information from each sentence, finally the output of CNN is fed to Long Short-Term Memory(LSTM) layer.

Due to time constraint, in this project our model architecture is different from theirs. We directly use already trained word2vec model on Wikipedia Text8 corpus to encode each word in the input tweets, instead of training the word2vec representation. There is no sentence composition, that is we do not apply CNN on each sentences in a tweet, the memory cell in LSTM layer is directly connected to each vector represented word in our model, and the final output is the predicted topic of that tweet, see Figure 4:

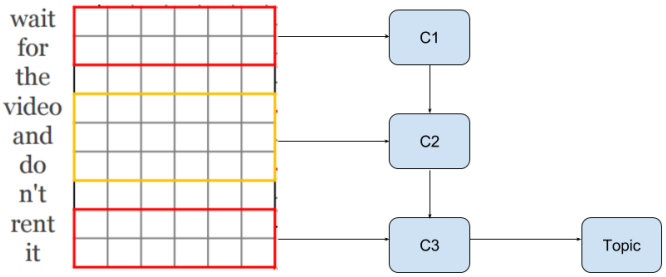

Fig. 4: Each word in sentence connects to a LSTM cell

Each of the memory cell contains 4 major operations,

- Forget gate: Provide a control over how much information come through from previous cell.
- Input node: Activation from input data at current time step.
- Internal gate: Update and provide different candidate.
- Ouput gate: Decide what to output and also what gets passed to next memory cell.

C. Build Model

1) Train: Combine all sports tweets collected into one data set, with total 44,755 number of tweets. Train the model with a stratified 5-fold cross validation, each with 75% training set and 25% validation set.

The size of hidden neuron is 300, number of memory cell is 23 which is our sentence length(number of words), dimension of each word is 600 as mentioned previously(so the total input data dimension is 23x600). We use batch size 50 and 20 epochs to train our model.

2) Validation: Prediction is made on the 25% validation set from previous split in total 5 times.

3) Result: The evaluation metrics of the result on validation set is *harsh accuracy* that is the mean of the accuracy per class.

Below is the output of the result from 5-fold cross valiation,

```
[CV 1]
Confusion matrix:
[[2335   35   28   64   27]
[  53 2356   52   78   42]
```

```
[  24   48 2289   56   16]
[  44   56   35 2342   32]
[  23   42   28   72 1012]]
(Harsh) Accuracy: 0.92

[CV 2]
Confusion matrix:
[[2427   52   26   37   36]
[  47 2378   37   29   34]
[  66   50 2281   31   18]
[  83   61   21 2239   78]
[  40   64   15   30 1009]]
(Harsh) Accuracy: 0.92

[CV 3]
Confusion matrix:
[[2387   46   24   90   20]
[  43 2278   30   76   47]
[  40   51 2293   51   18]
[  26   44   21 2392   34]
[  39   38   18   72 1011]]
(Harsh) Accuracy: 0.92

[CV 4]
Confusion matrix:
[[2386   58   17   46   15]
[  64 2311   27   63   21]
[  63   42 2339   39   14]
[  68   44   27 2291   33]
[  76   72   25   92  956]]
(Harsh) Accuracy: 0.91

[CV 5]
Confusion matrix:
[[2239   38   46   78   39]
[  28 2295   47   55   49]
[  20   36 2442   43   13]
[  33   35   33 2383   42]
[  27   42   18   74 1034]]
(Harsh) Accuracy: 0.92
```

We see that it achieves remarkble result on this 5-classes sports classification task, it has high harsh accuracy and low variance.

VIII. CONCLUSION

...

REFERENCES

[1] J. Li, H. Xu, X. He, J. Deng and X. Sun. Tweet Modeling with LSTM Recurrent Neural Networks for Hashtag Recommendation *International Joint Conference on Neural Networks (IJCNN)*, 2016

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *In Proceedings of NIPS*, 2013.