

Twitter Topic Recommendation with a Deep Learning Model

Names

Abstract—Need to write

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 et al [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] 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 et al [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 et al [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 falls in, 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:

- Replace http links with whitespace.
- Replace hashtags {#basketball, #hockey, #baseball, #volleyball, #tennis} with whitespace.
- Replace newline characters such as \r, \n and \r\n with whitespace.
- Replace special symbols such as #, @, %, &, \$, digits and punctuations with whitespace.
- Remove ‘RT’s (retweets).
- Combine multiple whitespaces to one.

- Remove duplicated tweets.

The order of $\{2,3\}$ and $\{4\}$ matters since if $\{4\}$ is executed first then $\{2,3\}$ would not work correctly. $\{1,5,7\}$ is 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. To explore whether we could achieve higher

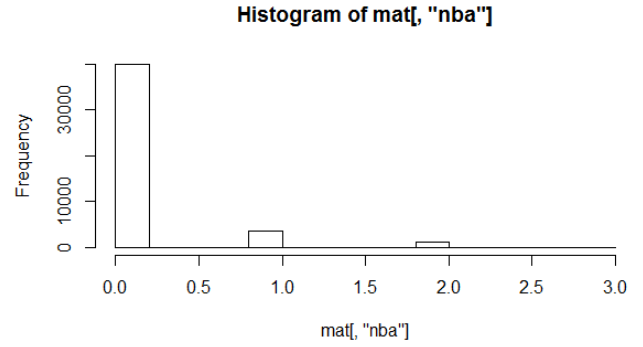


Fig. 1: Non-Gaussian distribution of NBA in document-term matrix

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 5.

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: Actual 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

VII. PREDICTIVE MODELING

A. Feature Engineering/extraction

Each data point in the raw input data 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 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 Fig 2.

Skip-gram models utilizes a shallow one-layer neural network to learn the representation of words such that the likelihood of obtaining surrounding words from a given words is maximized. In this project, we mapped each unique words into

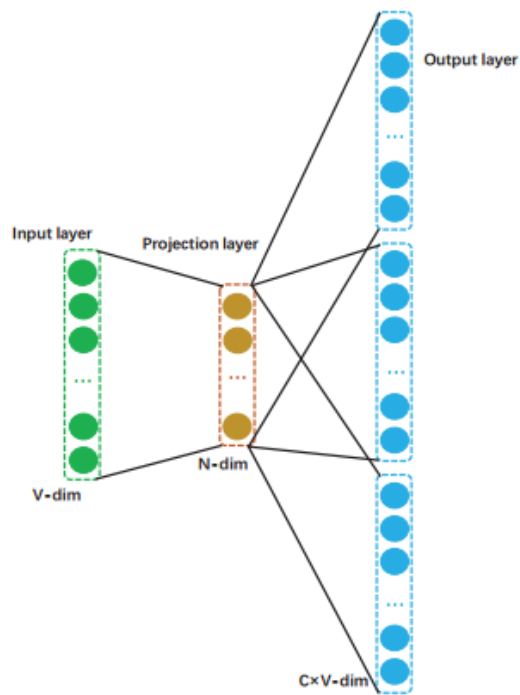


Fig. 2: The skip-gram model architecture [1].

a 600-d low dimensional vector space, i.e. the number of features used in our project are 600. We could convert the data to even higher feature spaces, but that would slow the training and not necessary as the model is already performing very well.

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.