Project Report

Analysing news and predicting number of Twitter shares

Venkatachalapathy Othisamy

Carlson School of Management, University of Minnesota

Case Report

Table of Contents

1. Business Understanding

1.1 Business Problem

How can New York Times (NYT) leverage text analytics to gauge popularity of a news article beforehand, for placing them in an effective manner, resulting in increased impressions on social network?

1.2 Motivation - To tying it back to the problem statement, I scoped the business problem to Twitter shares. This was a crucial decision, as Twitter acts as the perfect platform for real time news, as and when it happens.

Moreover, unlike other social platforms, Twitter has a large base of famous people from government heads to celebrities to decision makers of highest levels. In addition, the news which are shared more could result in widespread awareness, resulting in better positioning of NYT brand.

Along with above said things, building a good brand name among Millennials is important for companies. Millennials would consider a news more important if it is shared on Twitter. New York Times wants to be equally popular among millennials, by knowing whether a particular news article is going to gain popularity on Twitter.

In a generation obsessed with being online, it is of paramount importance that New York Times leverages text analytics to gain an edge over other brands which are more popular on Twitter.

2. Data Understanding

2.1 Data Source

New York Times has APIs using which I can get the required data for performing analysis and for building the model. I collected this data in text format from: https://developer.nytimes.com/.

The data was received in two files. One file contains attributes like URL, snippet, abstract, word count, published date, Twitter shares and more. The second file contains the URL and the body of the news(text). I merged these two files on the basis of URL of each article to make a final dataset to be used for text analysis. The merged file has **89810** observations containing unique news articles.

2.2 Data Types

I also classified the type of data I had in each column.

Our data contains three types of data(variables) which are described as below: -

1. **Nominal(Categorical):** A categorical variable (sometimes called a nominal variable) is one that has two or more categories, but there is no intrinsic ordering to the categories. There is no numerical significance of the values.
2. **Ordinal:** An ordinal variable is similar to a categorical variable. The difference between the two is that there is a clear ordering of the variables.
3. **Interval:** An interval variable is similar to an ordinal variable, except that the intervals between the values of the interval variable are equally spaced.

2.3 Dataset Description

Below is a brief description of columns in the dataset: -

Variable	Type of Data	Description	Sample Data
URL	Nominal	URL of news article	http://6thfloor.blogs.nytimes.com/2014/01/23/how-our-hillary-clinton-cover-came-about/
Date_Collected	Ordinal	Date on which news article was extracted	14:30.3
Snippet	Nominal	Small piece of brief extract	The evolution of "Planet Hillary.
Abstract	Nominal	Summary of news article	How Our Hillary Clinton Cover Came About
Pub_Date	Ordinal	Date on which article was published	54:11.0
Word_Count	Interval	Count of words in the news	567
Twitter(Target variable)	Interval	Number of shares on Twitter	768
Nyt_news	Nominal	Body(Text) of news article	When we created the cover of this Sunday's magazine to accompany Amy Chozick's article about Hillary Rodham

Below is a summary statistics of the columns: -

```
    URL            date_collected      snippet            abstract          pub_date          word_count
Length:90316       Length:90316       Length:90316       Length:90316       Length:90316       Length:90316
Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
   twitter            news
Min.   :    0.0   Length:90316
1st Qu.:    0.0   Class :character
Median :   54.0   Mode  :character
Mean   :  235.1
3rd Qu.:  210.0
Max.   :73433.0
```

3. Data Preparation

3.1 Data cleaning

After merging the two files, I had the task of making the dataset fit for performing modelling operations on it.

To clean the data, I performed the below steps: -

- Removed the rows where there were NULL values or blanks cells in Nyt_news column and snippet columns as I wouldn't be able to analyze these rows. These rows were incomplete set and unfit for performing analysis.

3.2 Choice of column for analysis

I considered snippet for performing text analytics for below reasons: -

1. **Brief replica of actual news article**

   Snippet is a brief replica of body of news and it contained all the important words that were present in the actual news article. So, analyzing snippet and making predictions on the basis of it made sense.

2. **Limited Computational Power**

   Due to limited computational power I had, processing huge amount of texts and applying complex models on it was computationally expensive.

3.3 Transforming text

I followed below steps to make the text at its best form to be used for analysis: -

1. **Converted characters to lowercase**

   Two words can be perceived differently by the model if the cases are different in them. To remove this possibility totally, I converted the characters of the snippet to lowercase.

2. **Removed punctuations**

Having punctuations at the start or end of a word confuses the models to treat them as different words. I removed punctuations from each snippet to have only proper words for analysis.

3. **Removed stop words**

   I removed stop words or the words which do not contribute to the generalization of the model. These words tend to be in almost all snippets and therefore, don't give any specific insights.

4. **Stemming of words**

   Tenses make the spelling of a word different in various forms. The meaning and sense of the word remains same. Like, run is the root word, ran, running are different forms of the same word. Models are naive to process them as different words. Therefore, I want to bring each word to be in its root form so, that the model evaluates each of them as one word.

3.4 Final transformation of data(tf-idf)

I am interested in knowing the significance of each word in predicting the popularity of articles. The current state of data is not appropriate to perform this analysis. For this reason, I needed to transform the dataset so that I have each word as a column and its frequency in each snippet. I transformed the data into a matrix of columns of words. However, considering all words that are possible in all of the snippets would potentially result in a huge matrix which would be difficult to process. After removing stop words, I still had some words which were not useful. [Example - good, bad - these words are very common but don't contribute much to the generalization performance]

For getting rid of these kind of words, I am using tf-idf transformation. In information retrieval, tf–idf, short for term frequency–inverse document frequency, is a numerical statistic

that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

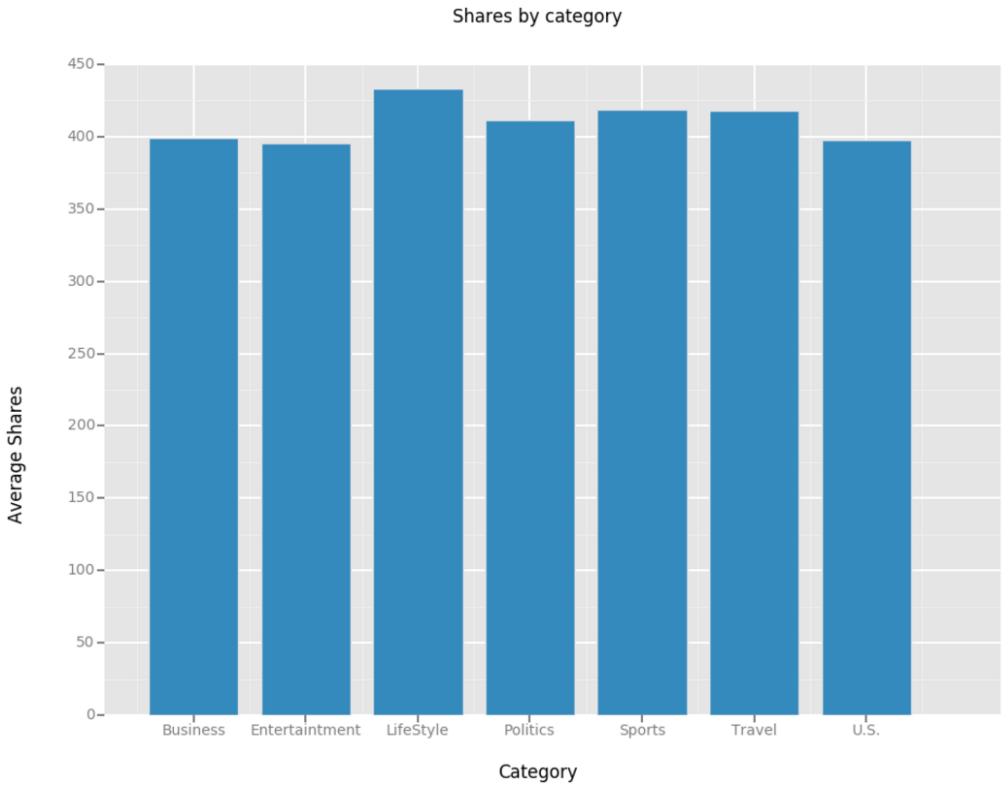
I built this term frequency - inverse document frequency (tf-idf) matrix, which transformed these words from specific articles as features and gave me their frequency as values in this cells of the table. However, as the news articles could be varying in length and contain diverse range of words, the matrix would be very sparse and wide. I further filtered out this matrix based on words that have a document frequency between a defined threshold (Maximum document frequency: 0.8, Minimum document frequency: 0.1) to remove words. After this step I will have the data matrix which I can then utilize to perform modelling.

3.5 Need for finding Category of news - Topic Modeling

News published on New York Times belong to different categories. Similar words in different categories have different prominence. Certain words will have more uniqueness in one category than the other. For eg., the word 'Business' in Business category is more common and does not add any value to the predictive power of the model. But the the same word 'Business' in other categories such as Politics or Sports will have more importance as it is not so common in those categories and similar words in those categories should be treated differently. New York Times doesn't save the information about category of new published articles in the database but this information can be very useful. The general model wouldn't be able to explain the variance of certain words for all categories. So, I would want to create separate models for each category. Therefore, I created one more attribute which defines the

category of the article. This has been done using Topic Modeling on the entire 'news' column using R.

As seen in the below graph, the most popular category as per average number of shares is Lifestyle news, followed by Sports.



LDA (Latent Dirichlet Allocation):

I prepared the original data to perform Topic Modelling on the news body by stemming, removing stop words, stripping whitespaces. Then I did content transformation on the corpus and created Document Term Matrix, which contained the frequency of the words in each of the documents. I used LDA(Latent Dirichlet Allocation) algorithm for performing Topic Modelling on the corpus. I used Gibbs Sampling method to create Document Term Matrix which gave me the topic distribution for each document. I then selected the topic which would have most likely produced that document.

In order to get the optimal number of categories for the entire corpus, I checked the entropy of topic distribution for each document. I used entropy, because it captures

ambiguity of the document topics. When the probability of a document belonging to two categories are almost equal, then the document is more ambiguous and the entropy will be high for that document. The higher the entropy in a document, more ambiguous the document's classes are.

The entropy value for most documents is lowest, when I split the corpus into 7 categories. Also, I found that NYT has 7 broad categories based on which the news items are divided.

Then, I gave names to the divided categories, by manually checking the most probable words in each category.

The retrieved Topics were related to

- Business
- Travel
- Style
- Sports
- Entertainment
- U.S.
- Politics

Below table shows the most probable words in each category that I created.

TOP 10 KEYWORDS	Business Topic 1	Travel Topic 2	LifeStyle Topic 3	Sports Topic 4	Entertainment Topic 5	U.S. Topic 6	Politics Topic 7
1	compani	citi	live	game	film	report	polit
2	year	restaur	famili	play	art	case	presid
3	million	car	kind	point	show	law	govern
4	market	island	peopl	score	theater	polic	nation
5	busi	year	time	team	music	court	support
6	bank	travel	work	year	perform	feder	unit
7	rate	hous	make	coach	director	health	trump
8	execut	day	thing	season	galleri	depart	american
9	increas	hotel	life	run	museum	offici	republican
10	price	move	school	leagu	center	investig	countri

3.6 Article's Age

One important feature of the data is age of the articles. There are articles which are just a day old at the time of extraction and then there are articles which are month old. These articles might have had similar characteristics and had been similar in content. However, due to the difference in age of the articles, these similar articles might have received completely different number of Twitter shares. Older articles would have received higher number of cumulative shares and a day old article would have received lesser number of shares. This could be simply because, many people did not get a chance to look at it.

I wanted to eliminate this age bias.

Therefore, I calculated the age of article by finding out the difference between the published date and extracted date. Then, I filtered the articles in every topic which have the age greater the mean age for that topic. By this way, this model will be trained better.

3.7 Popular articles:

The next important consideration is to filter out the articles which are more popular. The business problem that I undertook is to give potentially popular news articles its right place in

NYT web page. Therefore, in order to better train this model, I filtered out the number of articles which were shared less than mean number of times for each topic.
This step also served to balance popular and unpopular articles, because, the number of articles which are really popular are very less when compared to the articles which have received zero or least amount of shares. To put it in numbers, the number of articles which received more than mean number of shares is seven times less than the number of articles whose shares are below mean.

4. Modeling

I used tf-idf matrix as predictors to predict the number of Twitter shares. I decided to build seven separate models for seven categories to better predict the popularity. The goal is to predict the number of Twitter shares. This is a regression problem. The options I had are:-

- Linear Regression
- Regression trees
- Support Vector Regressor
- k-NN Regressor

The final model was built by using Support Vector Regressor, because of the following reasons: -

1. **Data Linearity:**

Support Vector machine runs Linear Regression in high dimensional feature space, and hence, the data can be non-linear in original feature space. Therefore, there is no need of making any assumptions about the data distribution. The reason being that SVM transforms the original data into higher dimension where the data becomes linear. SVM then applies

linear regression on the higher dimensional data, the objective function of which is to reduce the residual error.

2. **Input parameters**

Another advantage of SVM algorithm is that it is not too dependent on too many parameters which can change the predictions considerably. I changed the penalty parameter(C=1) and the kernel function(rbf) to get the optimum predictions.

3. **Handling high-dimensional data**

The another reason on why SVM was chosen is that, it can handle high dimensional data and there is no limit on the number of attributes that can be given to the algorithm. The text data will have large number of features, as every word is considered a feature, which SVM can handle well.

Alternatives:

Below are some of the alternative algorithms that I tried: -

- Neural Networks (MLP regressor)

I also tried Neural Networks, since it had the inherent ability to detect non-linear data and hence, I need not make any assumptions about the relationship between target variable and the response variables. But Neural Networks did not perform satisfactorily, probably because the training size was not sufficient for Neural Networks.

Also, the choice of Neural Networks for the business problem in hand may not be good. This is because the author of a news article may use the model multiple times to check whether the news that he is giving will be popular or not. Since, Neural Networks takes too long to train, the choice may not be good, as the stakeholders might want results in real time.

- k-NN regressor

k-NN regressor did not give good results because this is a high dimensional data, and the data are distributed way too far in those dimensions and hence calculating distance metrics for the prediction might not be meaningful. As expected, all the models which used distance based metrics did not give me the good results.

- Regression trees

Regression trees performed similarly to k-NN regressor but could not beat the performance of SVMs. One reason might be that the Regression trees assume linear relationship between the predictors and the response variable. Since the data is non-linear, the Regression trees did not perform well.

Model Assumptions:

- Context of sentences do not influence the popularity of the article.
- Order of words do not matter.
- Older articles will have more shares than similar recently published articles
- I assumed an article it is shared more number of times than the mean shares for that category.

5. Evaluation

I evaluated the models using MAPE.

Mean Absolute Percentage Error (M.A.P.E) - It measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error. This measure gives me the deviation from actual in percentage terms so, it makes more sense to use it to measure the performance of this model.

Also, I can compare the performance of my model among different categories.

Models in each category have performed as below: -

M.A.P.E

Category	MAPE
Business	: 40.14%
Travel	: 37.17%
LifeStyle	: 38.42%
Sports	: 39.31%
Entertaintment	: 38.31%
U.S.	: 38.90%
Politics	: 38.29%