Generate Sentiment Polarity of Tweets Using LSTM with Word2Vec/GloVe

Wei Yao
SFU
yaoweiy@sfu.ca

Nicholas Carr
SFU
ncarr@sfu.ca

Abstract

We generated a model to classify the sentiment of tweets into polarity as either positive or negative. Which is the phase one version of our project and is supposed to be the baseline of our phase two. For some reason, we only accomplished phase I, mainly due to the inaction of the member Nicholas. According to our previous goal, we proposed an approach to build a model based upon phase one which would automatically generate graph-based hashtag for each tweet even if there is none or multiple original hashtags. As we know, grouping similar content, especially on Twitter, is primarily determined by the hashtags included by the users, and it's often misleading. Our proposed approach would fix the issue and give the users more helpful information when they search for some interesting topics. We trained our model with millions of accurate and balanced data and our model reaches a standard of accuracy over 0.80 which would give out meaningful feedback of the polarity of a single tweet(or any other text messages) in most cases. We achieved this task by experimenting with a sequential LSTM model with two different embedding techniques, that is, Word2Vec and GloVe. This paper also describes the preprocessing steps of data along with optimizers of model parameters needed to achieve high accuracy.

1 Introduction

As posts are growing rapidly online on the social media platform, Hashtags play an important part in the user experience, especially in Twitter. Hashtags being listed as trending can draw a great deal of attention from the greater Twitter community, however, the naive approach of simply grouping all tweets with the same hashtag will yield potentially misleading results. Hashtags are unable to take into account the context of the tweet in which they occur, such as if a tweet is intended as a meta-commentary on the hashtag. In which this issue became our motivation to do this project. Our interest is to develop a robust classifier for tweets based on the sentiment analysis in language content. Our primary goal is to maximize the accuracy of our classifier, using different neural net architectures, word embeddings, and the inclusion or exclusion of certain types of data (e.g emojis, embedded media, URLs). Our hypothesis is that we can generate a model with high accuracy on classifying sentiment in Twitter messages using deep learning techniques with Word2Vec and Glove. To do so, we conducted some research about what we can do and what we should do to obtain our data and do the pre-processing work. We tried to do it by ourselves using an automatic tweets extractor build from scratch while the generated dataset is not ideal for our model, which was abandoned and is considered to be the experiment dataset of our further work. For instance, we can extract millions of tweets with query ['COVVID-19' , 'vaccine'] and fed into our sentiment analysis model to see the attitude of the public towards the upcoming vaccine. Also, we learned several papers besides our lectures about Word2Vec and Glove to understand the logic behind to better implement our model with these two embedding strategies. And we achieved the goal with accuracy over 0.80 for both of them together with the LSTM deep learning pipeline. What's more, the importance of doing this type of sentiment analysis is that it would be useful for people to do any topic of research on social media, such as a review of a movie, products, services of a company without reading through the whole long text. People just need to click the button and the model would give them an answer whether the whole message is saying positive or negative. And if we can finish the phase two version, it would be even better in terms

of accuracy and convenience, especially for Twitter users.

2 Prior Related Work

During our work of research, we found that the community has done a large amount of excellent worked on detecting sentiment in text. And most of which is focused on classifying larger pieces of text such as reviews. Turney presents a simple algorithm called semantic orientation for detecting sentiment in 2002.[1] Pang and Lee present a hierarchical scheme in which text is first classified as containing sentiment, and then classified as positive or negative.[2] What's more, Pang et al. have also analyzed the performance of different classifiers on movie reviews.[3] The work of Pang et al. is dramatically important to the community which is treated as the baseline to conduct further work across different domains in the field of natural language processing. In addition, the work done by Alec et al from stanford inspires us mostly in this project, where they introduced a novel approach to do sentiment analysis with distant supervision.[4]

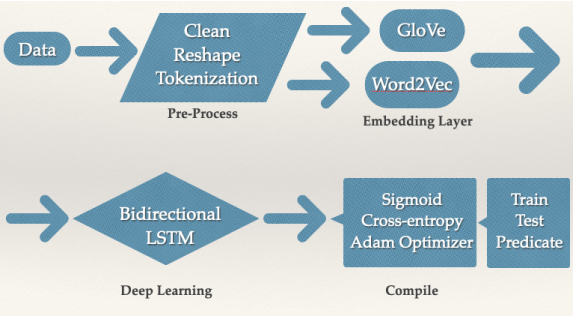
3 Model and Approach

3.1 Original Model

Before the shift of our goal, our main approach would be a combination of sentiment analysis of tweets based on its own Hashtag and the generation of new Tags as a candidate based on context level word embeddings to form a powerful classifier. And then people can use this classifier to better search posts/tweets based on their interests rather than only look at those tweets with the same Hashtags. We tried to build some by ourselves but the overall performance and flexibility is not satisfactory. After the pre-classification for all tweets with the same Hashtag into two different groups: positive, negative. We use the technique of deep learning to perform final classification by generating a final candidate as the word of the main tag of the tweet(if its score exceeds the original Hashtag). In order to accomplish this task, we would convert word embeddings into the forms of GloVe (including both pre-trained embeddings and learned embeddings), and use Pytorch for various deep learning architectures.

3.2 Current Model

However, after we narrowed down our goal onto phase one only, our current model consists of several components as shown below:



First, we perform the pre-process for the input data by doing clean, reshape, and tokenization work. Then two divergent embedding layers are applied separately which are Word2Vec and GloVe where words are converted to representational vectors. Next, we use the sequential LSTM together with the current embedding layer and apply cross-entropy with Adam optimizer. Finally, feed them into a sigmoid layer and compile the whole model.What we can obtain through this approach is a model ready to predict the sentiment polarity of any text message, especially towards tweets since which share the same domain as our training data. Here is the summary of the model with Word2vec embedding:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 300, 300)          196422900
_________________________________________________________________
dropout (Dropout)            (None, 300, 300)          0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               160400
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 196,583,401
Trainable params: 160,501
Non-trainable params: 196,422,900
_________________________________________________________________
```

3.3 Word2Vec

Word embedding is considered as one of the most popular representation of document vocabulary neural network developed by Tomas Mikolov in 2013 at Google. Which can be used to capture the context of a word in a text, semantic and syntactic similarity, relation with other words, etc. We can use Skip Gram and Common Bag Of Words (CBOW) to implement the Word2Vec while both involving neural networks.

3.4 GloVe

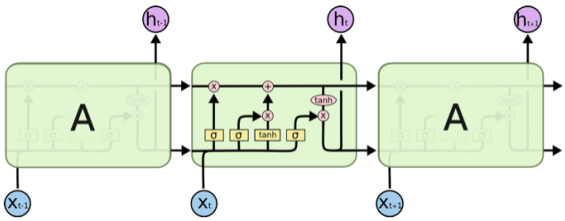
GloVe is a variation of a Word2Vec model which is designed as an unsupervised learning algorithm to obtain vector representations of words. We can treate GloVe embeddings similarly to pre-trained AlexNet weights. For example, consider the co-occurrence probabilities for target words

ice and steam table presented from stanford[5]:

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

3.5 LSTM

Long Short Term Memory networks(LSTM) are RNNs that can handle the task of learning long-term dependencies. LSTM is later refined and popularized by many contributors in the community, which was firstly introduced by Hochreiter and Schmidhuber (1997)[6]. The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along with it without any modification. Here is a diagram representing the details of its module[6]:



3.6 Sigmoid

Sigmoid activation function, sigmoid(x) = 1 / (1 + exp(-x)).

3.7 Maximum Entropy

Maximum entropy states that the probability distribution which best represents the current state of knowledge is the one with largest entropy,. The model is represented by the following:

$$P_{ME}(c|d,\lambda) = \frac{\exp[\Sigma_i \lambda_i f_i(c,d)]}{\Sigma_{c'} \exp[\Sigma_i \lambda_i f_i(c,d)]}$$

where c is the class, d is the tweet, and λ is a weight vector.

3.8 Adam Optimizer

Adam stands for adaptive moment estimation. Briefly, this method combines momentum and RMSprop (root mean squared prop) by introduces four hyper parameters:

- learning rate alpha
- beta from momentum
- beta2 from RMSprop
- epsilon

Some advantages of Adam include:

- Relatively low memory requirements (though higher than gradient descent and gradient descent with momentum)
- Usually works well even with little tuning of hyperparameters.

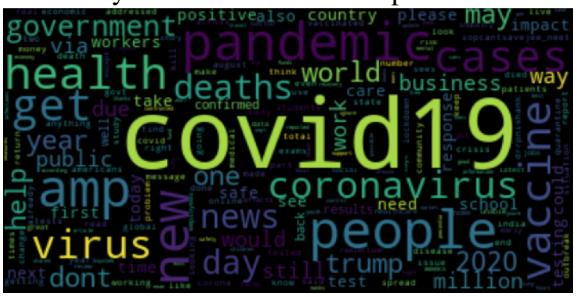
4 Data

4.1 Original Extractor Data- Abandoned

To improve our model accuracy and reliability, we build the tweet extractor using REST API and output our queried tweets into the D2.txt file. Then we implemented several helper functions mainly focused on remove(url + punctuation + stop words) and split them into lists of tokens of each tweets by our knowledge from regular expression lesson. And we successfully extracted most 100 frequented tokens in those tweets as:

[('covid19', 657), ('people', 78), ('new', 78), ('amp', 76), ('pandemic', 75), ('health', 56), ('cases', 52), ('get', 52), ('vaccine', 50), ('virus', 45), ('coronavirus', 42), ('deaths', 36), ('news', 33), ('day', 33), ('government', 31), ('world', 29), ('may', 29), ('dont', 28), ('year', 28), ('2020', 28), ('one', 27), ('trump', 27), ('help', 26), ('public', 26), ('way', 26), ('business', 25), ('million', 25), ('still', 24), ('via', 24), ('would', 23), ('due', 22), ('work', 22), ('positive', 22), ('could', 21), ('see', 21), ('country', 21), ('today', 21), ('need', 20), ('impact', 20), ('please', 20), ('first', 20), ('time', 20), ('test', 19), ('safe', 19), ('school', 19), ('workers', 19), ('next', 19), ('response', 18), ('take', 18), ('care', 18), ('testing', 18), ('also', 18), ('confirmed', 18), ('back', 18), ('well', 18), ('results', 17), ('sopcantsavejee_neet', 17), ('lockdown', 17), ('august', 17), ('live', 17), ('total', 17), ('look', 17), ('working', 16), ('corona', 16), ('said', 16), ('state', 16), ('money', 16), ('students', 16), ('kill', 16), ('patients', 15), ('read', 15), ('exams', 15), ('covid', 15), ('know', 15), ('spread', 15), ('economic', 15), ('even', 15), ('message', 15), ('tested', 15), ('india', 15), ('anything', 15), ('issue', 14), ('think', 14), ('quarantine', 14), ('crisis', 14), ('number', 14), ('getting', 14), ('global', 14), ('community', 14), ('right', 14), ('many', 14), ('years', 14), ('find', 14), ('already', 14), ('use', 14), ('made', 14), ('addressed', 14), ('americans', 14), ('times', 13), ('online', 13)]

To get better preparation of our final classification, we use the WordCloud tool to generate a clear view of those frequent tokens as below, however we found that words such as flu and cold doesn't show at the centre region of the word cloud. Which proved our assumption that Hashtag would not always reveal the main topic of a tweet:



For the dataset, we tried to extract tweets based on different queries/Hashtags to improve the quality of our dataset. Also, for the purpose of this project, we collected tweets consist of English only to better capture the idea of classification. Our tweets extractor was designed to make it powerful enough to collect high-quality data for our later model training and testing. However, we failed to obtain the same achievement in the middle of the project. So we abandoned those data and tuned onto an online open-source with another dataset to do our later work, which is from Stanford, and details are included below.

4.2 Current Input Data

The data is a csv file with emoticons removed which has 6 fields/columns from Stanford[7]:

0 - the polarity of the tweet (0 = negative, 4 = positive)

1 - the id of the tweet

2 - the date of the tweet

3 - the query

4 - the username of who posted the tweet

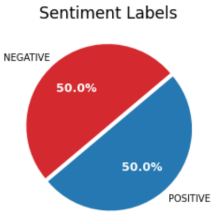
5 - the text of the tweet

Here is a table of head(10) data after several clean and filter work done:

	target	ids	date	flag	user	text
0	NEGATIVE	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	switchfoot awww thats a bummer you shoulda ...
1	NEGATIVE	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he cant update his facebook by t...
2	NEGATIVE	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	kenichan i dived many times for the ball manag...
3	NEGATIVE	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	NEGATIVE	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	nationwideclass no its not behaving at all im ...
5	NEGATIVE	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	kwesidei not the whole crew
6	NEGATIVE	1467811592	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	mybirch	need a hug
7	NEGATIVE	1467811594	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	coZZ	loltrish hey long time no see yes rains a bit...
8	NEGATIVE	1467811795	Mon Apr 06 22:20:05 PDT 2009	NO_QUERY	2Hood4Hollywood	tatiana_k nope they didnt have it
9	NEGATIVE	1467812025	Mon Apr 06 22:20:09 PDT 2009	NO_QUERY	mimismo	twittera que me muera

The reason why we chose this dataset as our input is that it has an accurate property in terms of sentiment polarity since its generated from a reliable system by Stanford[6] and it's balanced in the distribution of data as we can see in the below pie chart:

```
Labels: ['NEGATIVE', 'POSITIVE']
Amounts: [800000, 800000]
```



The whole data consists of half positive and half negative samples, what's more, it has large enough amount of data size for us to train and boost the overall accuracy. Each label has 800,000 tweets to be trained, all data was cleaned out with emojis.

5 Experiments

We carry out our experiment based on our approach workflow. And we separate the experiment into two ways as illustrated below.

5.1 Embedding with Word2Vec

We follow the pipeline to tokenize text, split data and Initialize w2v model and build vcab of the w2v model. After 32 epochs of training, we compute the embedding matrix from w2v model.wv ['word'] and we successfully obtain the embedding layer now. So we add the sequential model with embedding layer, dropout ratio, LSTM model and the sigmoid and we have the entire sentiment analysis model. After several try of modifying parameters of the model, we pick the number 8 as our epochs and validation split to be 0.1. As we found the trade-off between increase of number of epochs is not worth trying both in running time and accuracy increase percentage(which is nearly none). In addition, since our PC is extremely slow in terms of CPU and GPU computations, we have only tried 3 times in total for the training. The most fast running time is still over 6 hours.

5.2 Embedding with GloVe

Then we almost repeat the work done in experiments as when we do with Word2Vec. There are several things that vary from what's been carried before due to the variation of GloVe compared to Word2Vec. First, we need to read glove vectors from the pre-trained glove file which trained on 6 Billion tokens and having a dictionary size of around 400 thousand unique words. Then we try the power of GloVe to exaim the similarity of a pair of two words:

```
: cosine_similarity(word_to_vec_map['cucumber'], word_to_vec_map['tomato'])
: 0.8632714714841137

: cosine_similarity(word_to_vec_map['cucumber'], word_to_vec_map['phone'])
: -0.012485506721235957
```

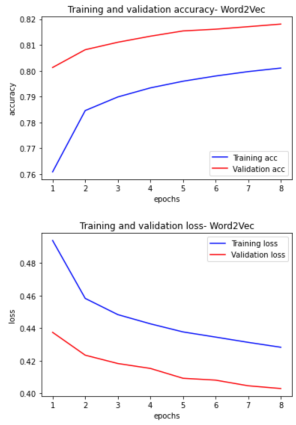
And then we build the sequential LSTM model again with GloVe this time and keep the same parameters such as epochs and dropout in order to compare the performance of two embedding layers. The running time of the model this time is much faster than the previous one.

6 Results and Evaluation

We will present the results in two situations as described below, separately in two different embedding strategies.

6.1 Word2Vec's model

After several attempts and improvements, both models with these two embedding layers work successfully, and the overall accuracy is not low even we have only trained for 8 epochs. And if we take a deep look of the plotted accuracy and loss(validation also) figure below:

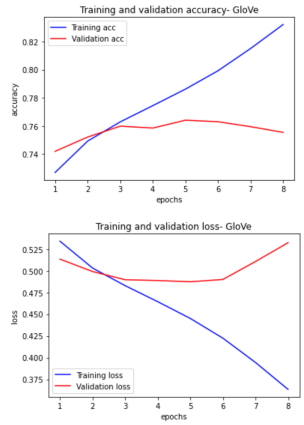


The model embedded with Word2Vec seems to achieve very good results as its continuously increasing in accuracy and validation accuracy. And there is no sign of overfitting as we can see the validation loss keeps dropping together with loss. Furthermore, the classification report tells us the confirmation that Word2Vec is doing well:

	precision	recall	f1-score	support
NEGATIVE	0.84	0.77	0.80	159494
NEUTRAL	0.00	0.00	0.00	0
POSITIVE	0.89	0.69	0.77	160506
accuracy			0.73	320000
macro avg	0.58	0.49	0.53	320000
weighted avg	0.86	0.73	0.79	320000

6.2 GloVe's model

However, the situation of the model with GloVe is not as good as in Word2Vec:



As we can see from the figure above, the validation accuracy stops increasing after the third epochs, which is a dangerous sign where overfitting might happen. And if we take a look at the loss figure, the validation loss stops dropping after the third epochs, which confirmes that the model is overfitted.

6.3 Application Test

The basic application of our model would be analysis the polarity of a single text, especially on tweets. Since tweets share the knoweledge domain as our training data, also which is our original goal to predict the sentiment of a tweet. We have evaluated some test cases in application as below:

```
predict("NLP is so interesting that I am getting obsessed")
```

```
{'label': 'POSITIVE',
 'score': 0.9262499809265137,
 'elapsed_time': 0.2972400188446045}
```

```
predict("I hate the winter")
```

```
{'label': 'NEGATIVE',
 'score': 0.013658285140991211,
 'elapsed_time': 0.07059192657470703}
```

```
predict("I don't like you")
# wrong here !
```

```
{'label': 'POSITIVE',
 'score': 0.9021734595298767,
 'elapsed_time': 0.07148313522338867}
```

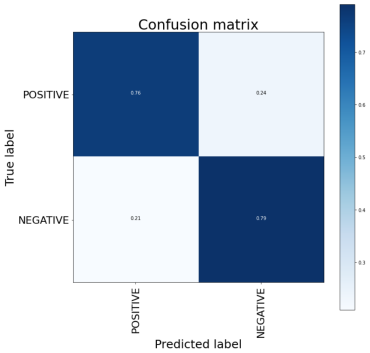
As we can see, the third case fails totally as our model can't understanding the trick that the 'don"t' word before the word 'like' would make the whole text to be negative in terms of sentiment.

7 Analysis & Conclusion

We perform the analysis mostly focused on where those models may fail. To better analyze, we generated the confusion matrix for Word2Vec, as well as the cloud-image of the most 20 similar words of 'good' obtained from this embedding. For Glove, we have tried several re-runs to see whether the overfitting is possible to eliminate, however, we fail to solve the problem, details also included below correspondingly.

7.1 Confusion Matrix of Word2Vec

Here is the confusion matrix of our model when embedded with Word2Vec:



The cases in positive and negative are mostly same as we can see from the figure above, which makes sense that our input traning data is balanced so that the model should have the similar performance to predict these two different labels.

7.2 Cloud-image of the Most 20 Similar Words

We extracted the top 20 words with the function most similar from the Word2Vec, and generated

the corresponding cloud-image as below:

As we can see, words such as 'rough', 'terrible', 'horrible' ... are all listed inside most similar words of 'good'. Which is spotted by us as the main issue cause Word2Vec to fail in its confused matrix.

7.3 Summary

According to the above evaluation and analysis carried in this project, we can draw some conclusions as well as some insights until this point although we have only accomplished phase one of our original goals. To conclude, both Word2Vec and GloVe is good enough to classify the sentiment polarity as either positive or negative for tweets and any other texts. The GloVe has the advantage of running/training speed and possible accuracy score, while it's easy to encounter the issue of overfitting. On the other hand, Word2Vec is steadily increasing in terms of accuracy but the training speed is dramatically slow which would take days if we run over 20 echoes with an indecent PC.

Based on our experience and knowledge, these two models won't be perfect if we don't add additional techniques to our model. Or maybe there is a chance if we can filter and clean our data even further. However, the ideas behind those two models are very intuitive, and it's easy for beginners to do some experiments with them. There is still some confusion for us why GloVe doesn't show the expected power which should be better than what Word2Vec can do. Since GloVe enforces the word vectors to capture sub-linear relationships in the vector space. Thus, it should perform better than Word2Vec in the task. There must be some reason why we fail to borrow the power of GloVe, although we noticed the issue of overfitting, which should not be the main issue.

8 Work Division

Wei Yao: here is a list of my contribution: Abstract: 50% - equally worked with Nicholas, writing words together in Discord

Proposal: 50% - equally worked with Nicholas, writing words together in Discord

Milstone: 100%

Project: 90% + did 90% work of research and 100% of coding

And since Nicholas didn't show up, I have nothing to present in his point of view although he did a lot in previous code assignments in our team. Also, I tried my best to deliver twice amount of work in order to fulfill the requirement of term project which should consists of more than two members. There is a lot space I can improve and contribute in the future.

9 Future Work

Our plan for the rest of the project would be mostly focused on using deep learning models to classify groups of tweets in positive, negative, and also neutral with a new tag(the main topic of the tweet). We are still reading related papers and exploring helpful projects in Kaggle in order to fulfill our classification model in phase two. We would adopt what we learned and implemented in homework4 as Parsing and contextual word embeddings into our model to boost the accuracy. Also, we would manually generate some test data by adding a new tag feature into each tweet since there is no good tool(with over 0.9 accuracies) we can use to generate automatically.

Also for the Glove embedding, there are usually over millions of different features which is a large number that ends in a high variance. So it will need much more training data to avoid overfitting. Our training set contains hundreds of thousands of sentences. But it is still a large number of features for our training set. It would be helpful if we can discard some useless features. Therefore, if we have time after we successfully implemented our entire model, we would try divergent ways to improve it as much as possible.

Furthermore, the overall experience we gain from this class is awesome, in terms of knowledge master and coding practice. We would like to explore deeper in NLP in our phase two version also in our future career.

References

[1] B. Pang, L. Lee, S. Vaithyanathan. Sentiment Classification using Machine Learning Techniques, 2002.

[2] B. Pang, L. Lee "Opinion Mining and Sentiment

Analysis" in Foundations and Trends in Information Retrieval, 2008.

[3] B. Pang, L. Lee, and S. Vaithyanathan. *Thumbs up? Sentiment classification using machine learning techniques*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 79–86, 2002.

[4] T. Sahni, C. Chandak, N. R. Chedeti and M. Singh. *Efficient Twitter sentiment classification using subjective distant supervision*. 2017 9th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, 2017, pp. 548-553, doi: 10.1109/COMSNETS.2017.7945451.

[5] GloVe: Global Vectors for Word Representation, `https://nlp.stanford.edu/projects/glove/`

[6] Understanding-LSTMs, `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

[7] Current data source link, `https://docs.google.com/file/d/0B04GJPshIjmPRnZManQwWEdTZjg/edit`