****

**Advances in Data Sciences and Architecture**

**Prof. Sri Krishnamurthy**

**Sentiment Analysis of Twitter Posts**

**Team 10**

**Xun Peng**

**Jiahua Liu**

**Task Description:**

Systems have to distinguish between Twitter posts that contain adverse drug reaction(ADR) mention versus those that do not.

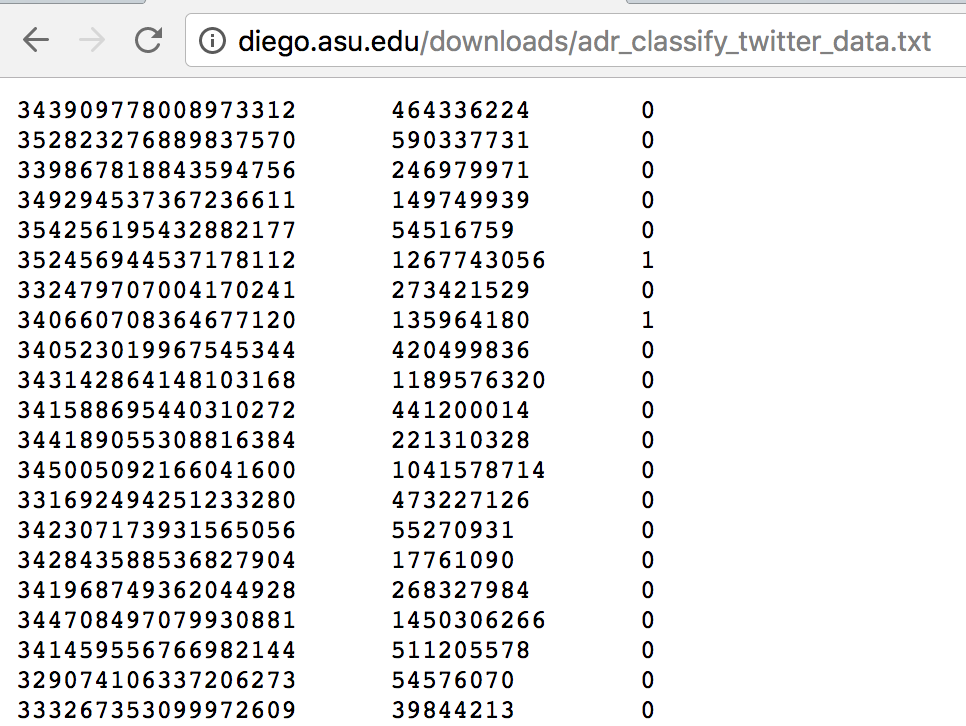
Automatic classification of adverse drug reaction(ADR) mentioning posts - binary classification

**1. DataSet**

We use crawler to fetch data based on Tweet ID and User ID.

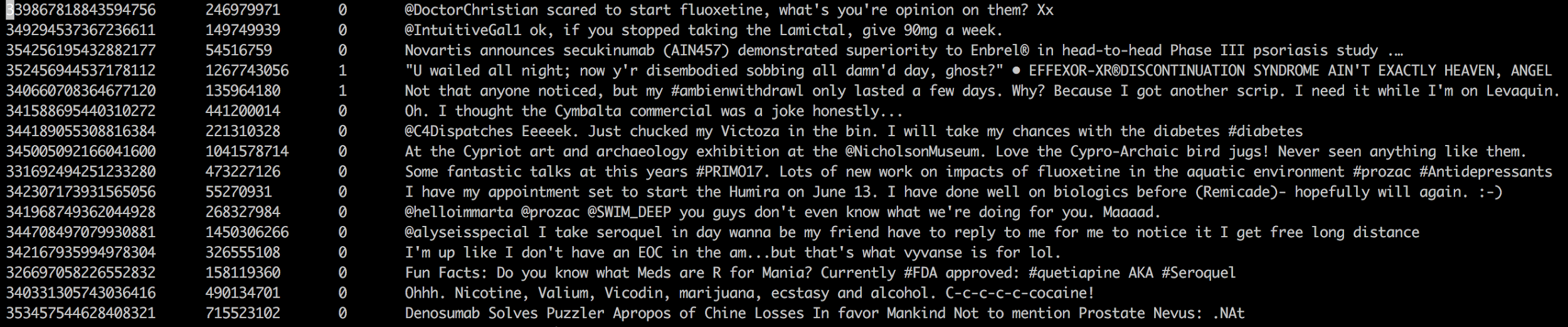
The base data link is belong:

<http://diego.asu.edu/downloads/adr_classify_twitter_data.txt>



These columns represent Tweet ID, UserID and the binary annotation indicating the presence or absence of ADRs.

We used a python crawler to fetch tweet post data from tweet and append this post data to the end of each row as a new column.



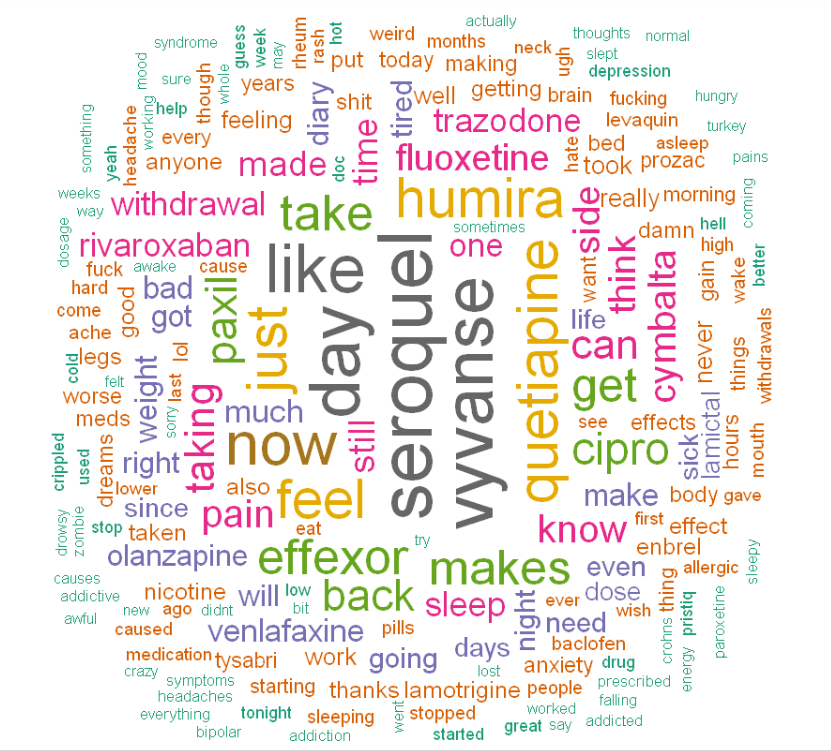
Data description:

1. Number of sentences: 6761
2. Number of sentences of label 0: 6034
3. Number of sentences of label 1: 727
4. Vocab size: 13657
5. Max sentence length: 40

**2. EDA**

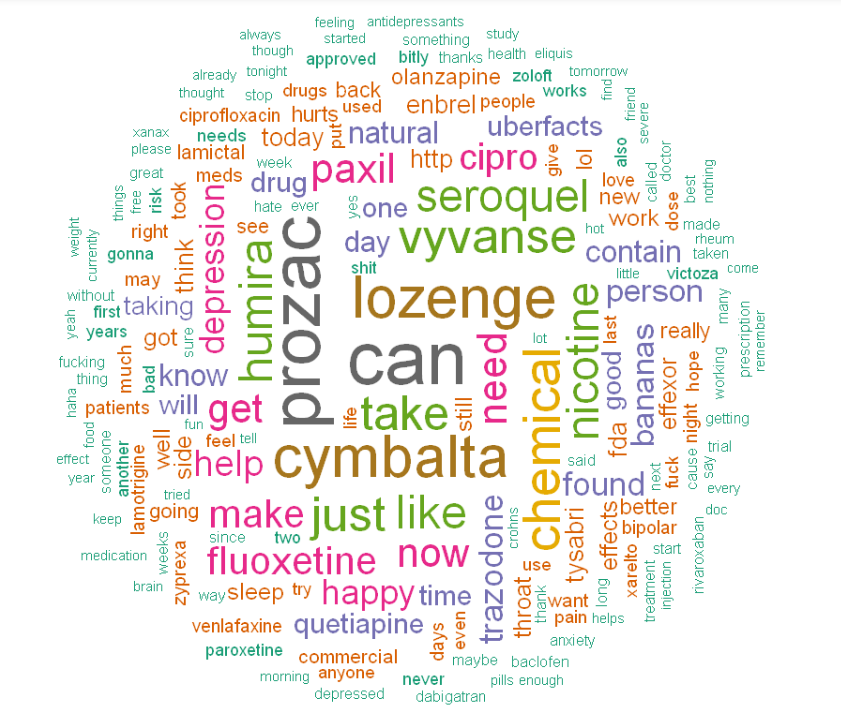
**2.1 Word Counts**

We analyzed the word counts in tweets that mention adverse drug reactions or not and look for patterns.



**Fig 2.1.1**

Fig 2.1.1 is the word cloud for tweets in which ADR information is present. The figure shows that there are some negative words that have high frequency in the corpus, such as pain, bad, worse, etc. We can also see lots of medical terminology in the center of the cloud, such as seroquel, vyvanse, humira, etc. From this figure we can get some basic idea about what people are more likely to talk about when they mention ADR in a tweet.

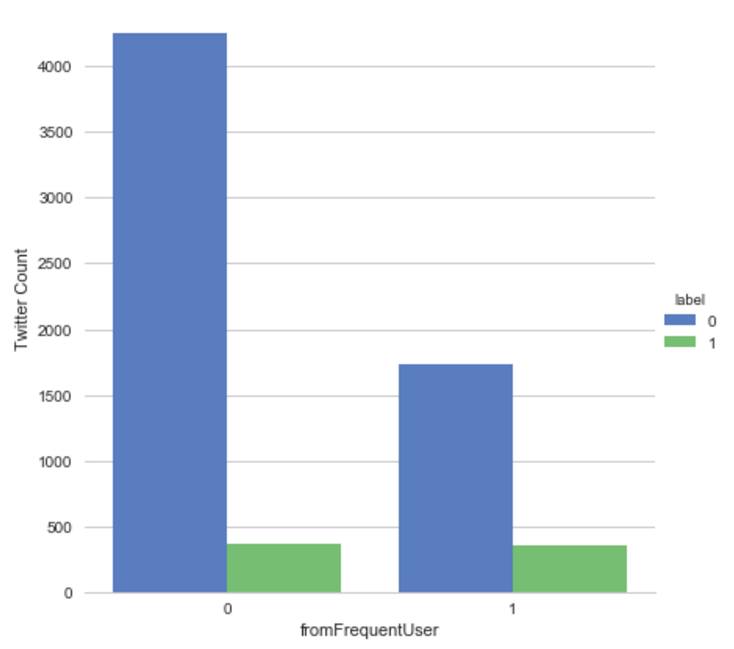


**Fig 2.1.2**

Fig 2.1.2 is the word cloud for tweets in which ADR information is absent. The figure shows that there are some positive words that have high frequency in the corpus, such as happy, well, etc. We can also see lots of medical terminology in the center of the cloud, such as lozenge, vyvanse, cymbalta, etc. From this figure we can get some basic idea about what people are more likely to talk about when they do not mention ADR in a tweet.

**2.2 User Frequency and ADR Presence**

We analyzed the correlation of users tweeting frequency and the presence or absence of ADR in a user’s tweets and looked for patterns.

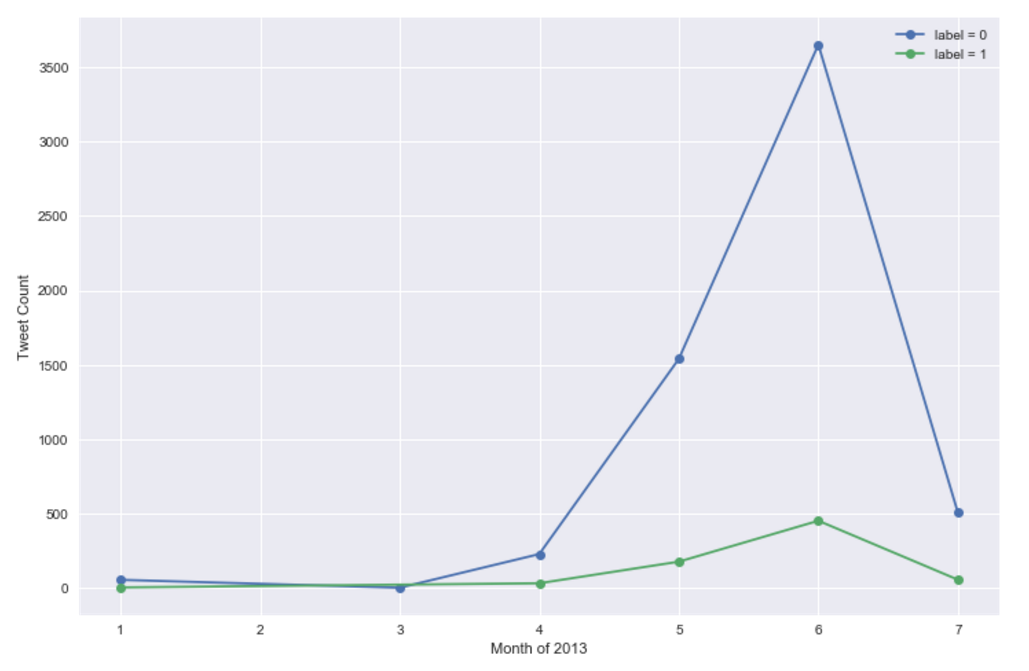


**Fig 2.2.1**

We categorized users who tweeted more than twice in our dataset as frequent users. From Fig 2.2.1 we can see that frequent users are more likely to mention ADR in their tweets, probably indicating that people who suffer from ADR tend to post more tweets than those who do not.

**2.3 Time-series Analysis**

We analyzed the ADR presence against time and looked for patterns.



**Fig 2.3.1**

Fig 2.3.1 displays the time-series trend of tweet both with and without ADR information. As our dataset contains tweets posted between January and July of 2013, data of that time period is shown on the plot. We can conclude that June saw the biggest number of tweets, both with or without ADR information, and that there is no obvious contrast between ADR presence and absence tweets in their trends in a specific month.

**3. Model Prediction**

**3.1 Data processing and data cleaning**

We first needed to split the dataset into two separate files based on their labels. While doing this part, we found that some Unicode characters (such as Unicode double quotes) caused problems in reading a file as a dataframe. We ended up reading the file as a plain text, replaced the Unicode characters with ANSI identical characters, and proceeded with the data processing.

**3.2 Distributed Representation:**

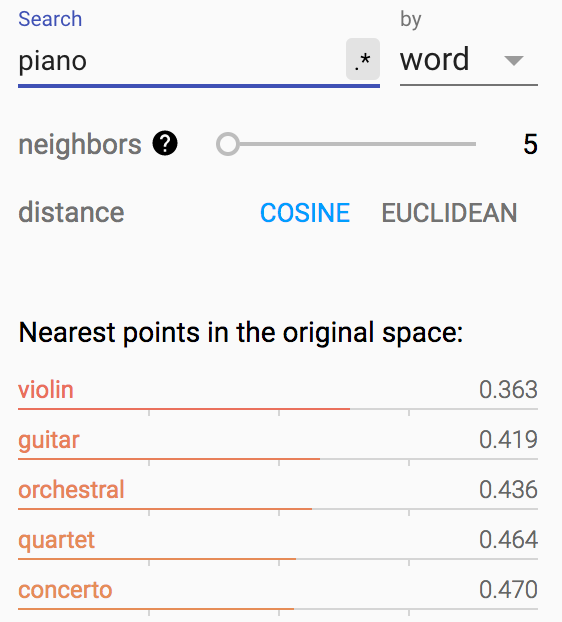
Our input are sentences. Distributed Representation is a way to train words to K-dimensional vectors. We can get the similarity of different words through the distance these their vector. We are going to use a tool called “word2vec” to train words. It was developed by researchers in Google AI research Group.

The link belong is the paper about word2vec:

<https://arxiv.org/pdf/1301.3781.pdf>



this figure represents vectors represent words

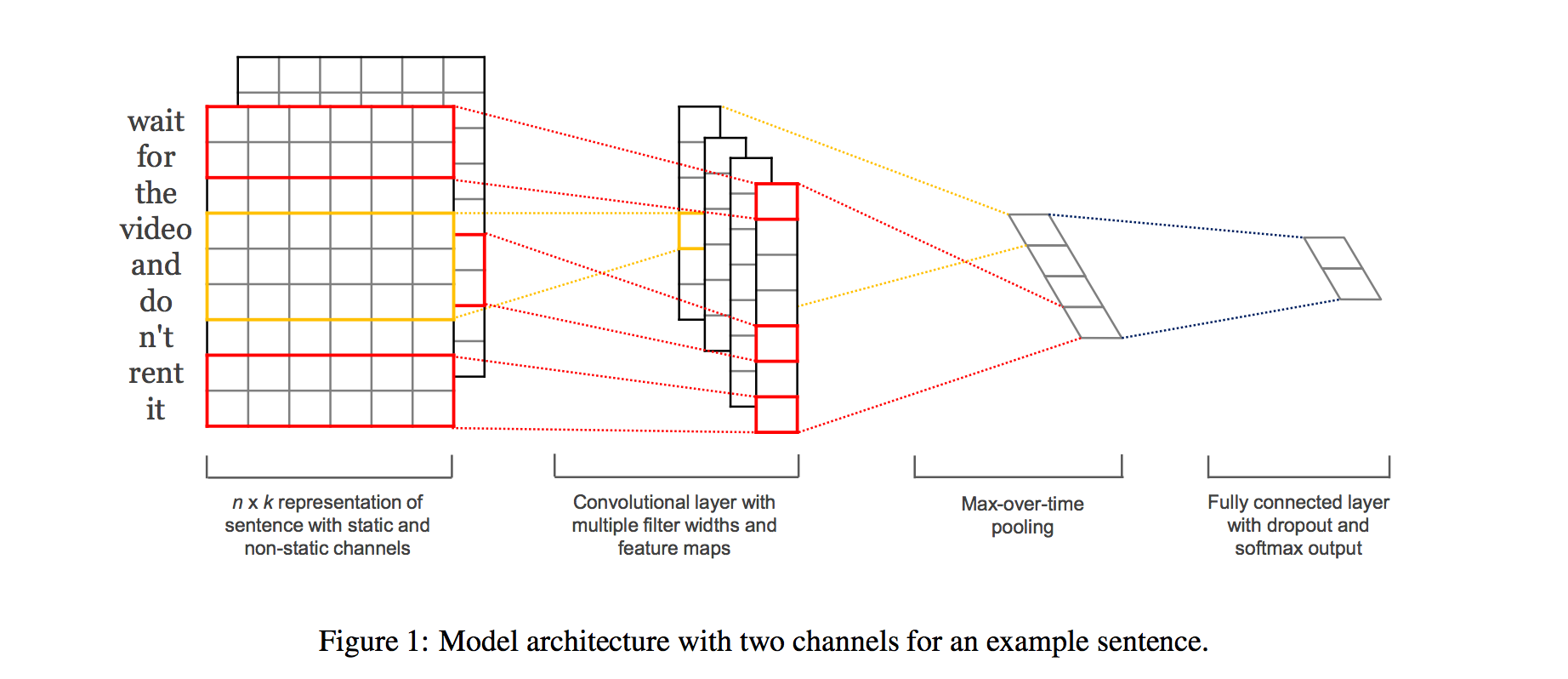


Eg:

vector(‘Paris’) - vector(‘France’) + vector(‘Italy’) = vector(‘Rome’)

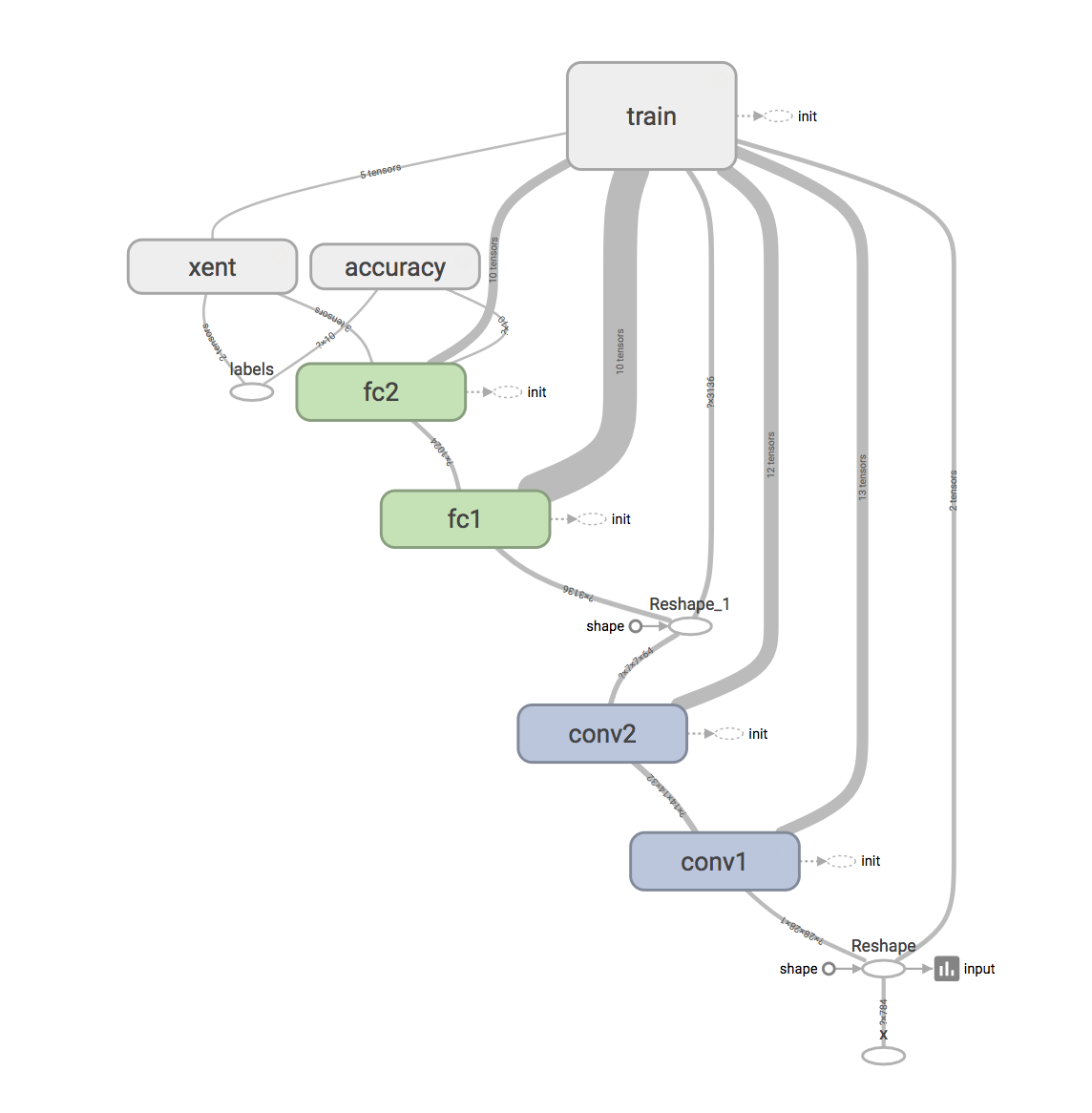
vector(‘king’) - vector(‘man’) + vector(‘woman’) = vector(‘queen’)

**3.3 Model**

For this part prediction model we are going to use TensorFlow to build Convolutional Neural Network(CNN) as our training model to predict Adverse Drug Reaction(ADR) mentioning in these post. 

**3.3.1 Convolutional Neural Network**

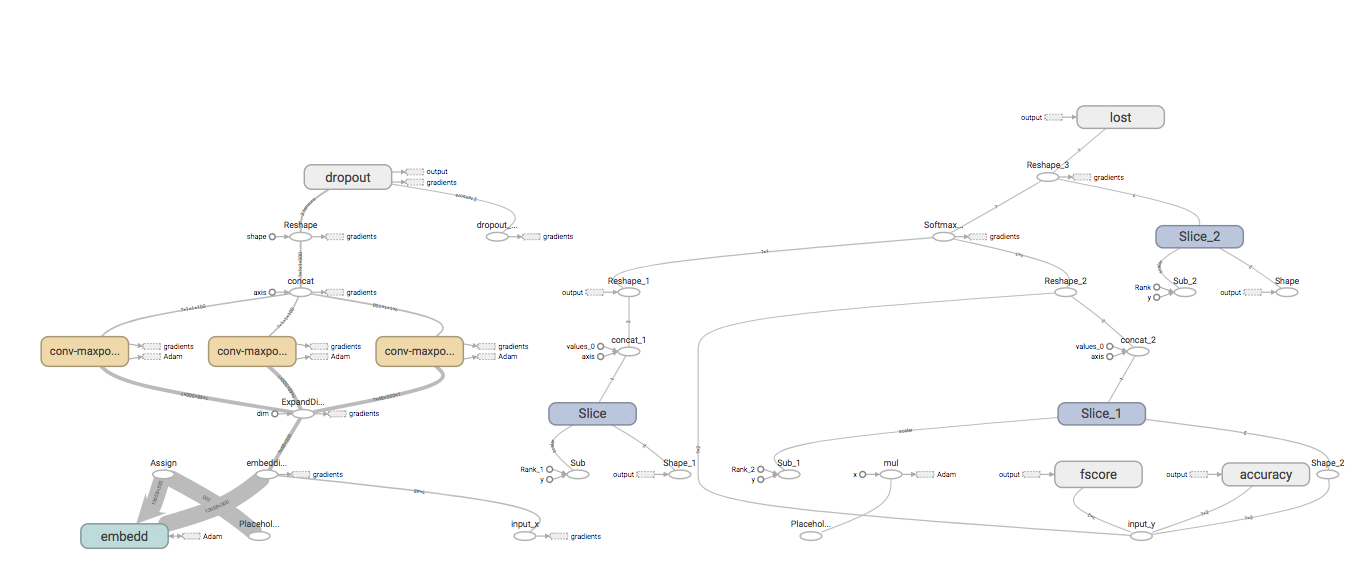
**1. Classical CNN on Computer Vision**



On My local system run

tensorboard --logdir=/tmp/mnist\_demo/7

This is a typical CNN with two convolution layer in MNIST dataset, two fully connected layer connect to convolution layer.

**2. Convolution Neural Network in NLP** 

**2.1 Build Model**

1. Embedding layer

1. Convert words to vectors based on word2vec

2. One channel

3. set embedding dimension to 300

2. Convolution layer and pooling for each filter

1. I use three different widths for filters [3, 4, 5] 100 for each filter

2. padded input to 48(vector\_length, max sentence length 40)

3. convolution operation based on filter and inputs from embedding layer

4. Maxpooling over the outputs of convolution layer

5. combine all the pooled features 3 types of filter and 100 filters for each type

3. add dropout

set dropout\_keep\_prob as 0.5 when we start train the model

4. set lost function

we use cross entropy to define lost function of this model

5. set optimizer

I use AdamOptimizer to minimize the lost function as we set the learning rate as 0.01 at beginning

**2.2 train model**

1. Convert each sentence into word vector with dimension of (48 \* 300) using word2vec

2. split all posts to two different files according to labels

3. split data to train dataset(90%, unbalanced) and test dataset(10%)

4. select same size of data from label 1 and label 0 to make a balanced training data. Repeat this process for 9 times(ratio of label0 to label1)

5. Set epoch as "25"

6. I use batch of size 100 to train model

**2.3 How to run model**

1. Clone the repository recursively to get all folder and subfolders

2. Download Google's word embeddings binary file from [https://code.google.com/p/word2vec/](https://code.google.com/archive/p/word2vec/](about:blank). Extract it, and place it under `data/` folder.

3. Using the command in the home directory of this repository

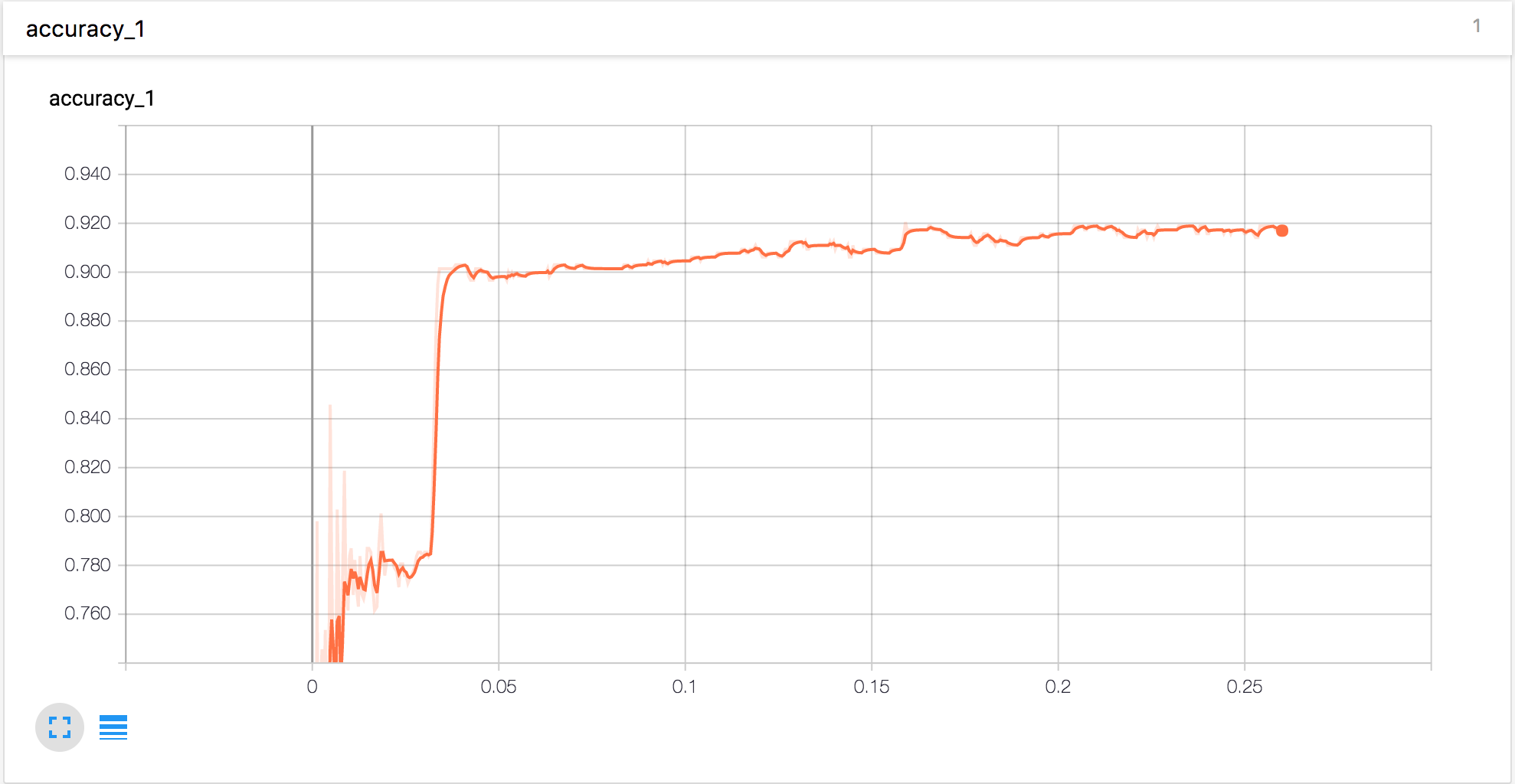
`python driver.py`

**3.3.2 TensorFlow**

Use command line below in terminal to run tensorboard for this model

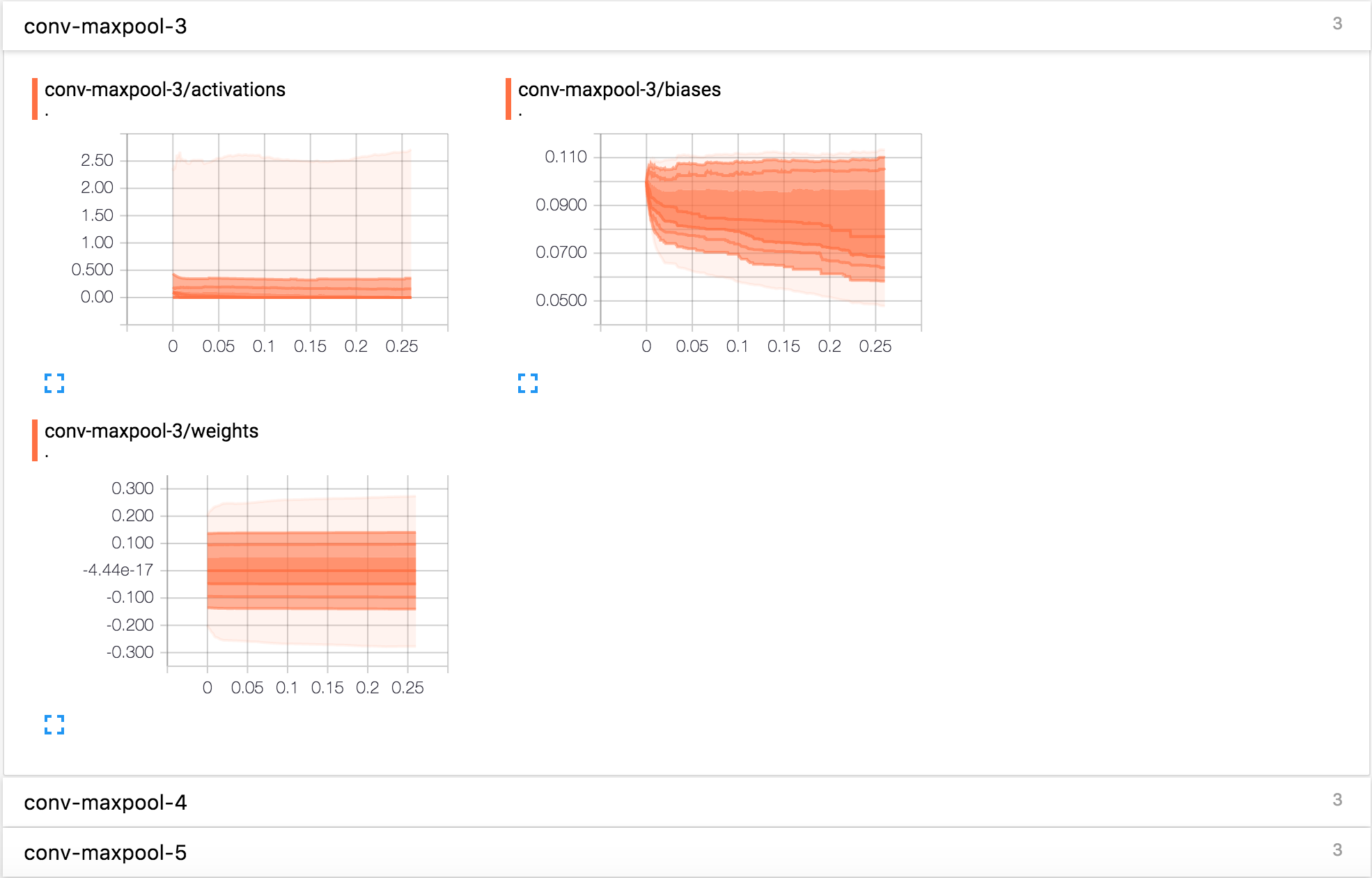
tensorboard --logdir=/tmp/tensorflow/1

Accuracy and lost function:



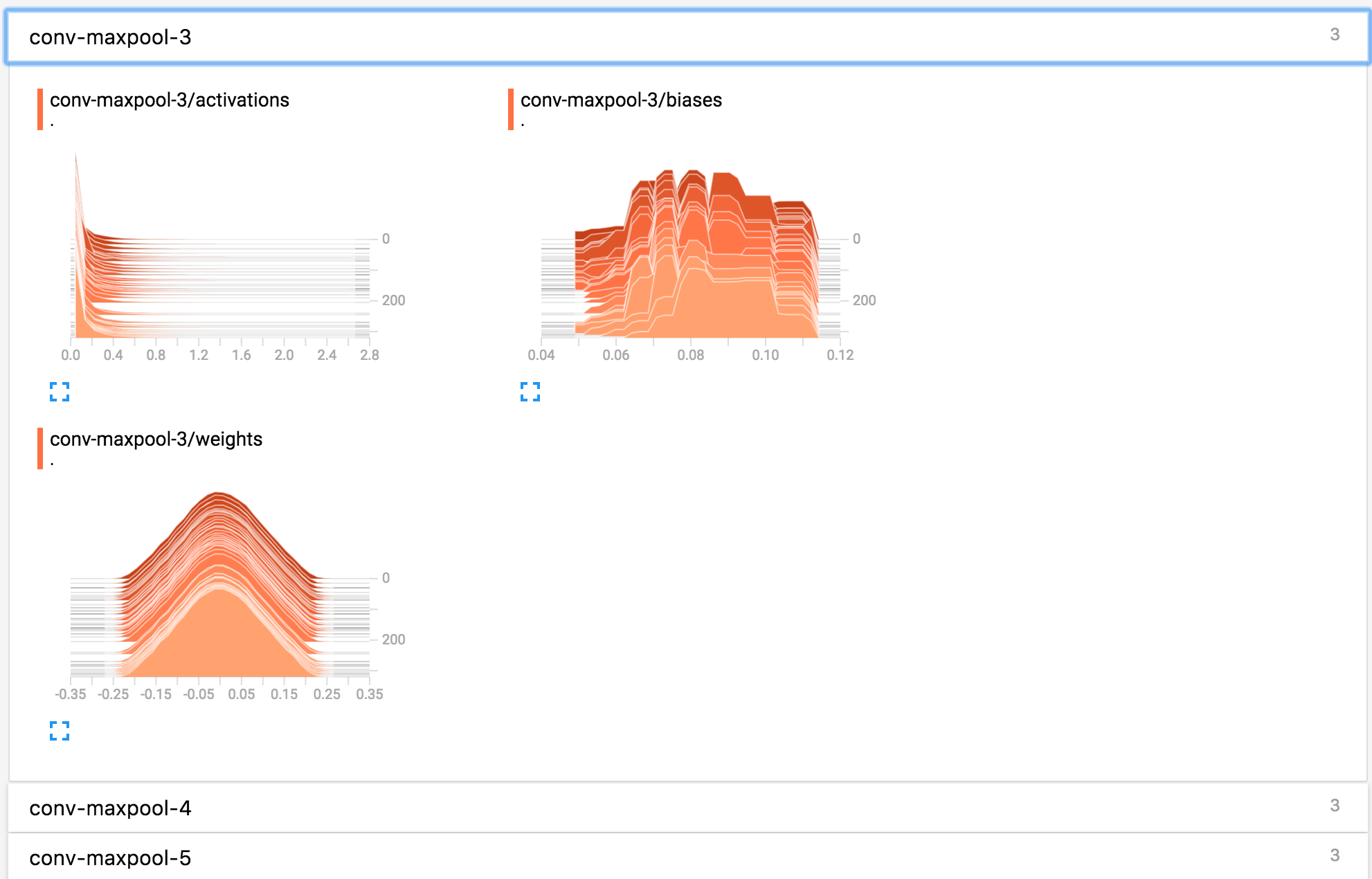
**In Distribution panel:**

The change of Activations, biases and weights of 3 kinds of filters

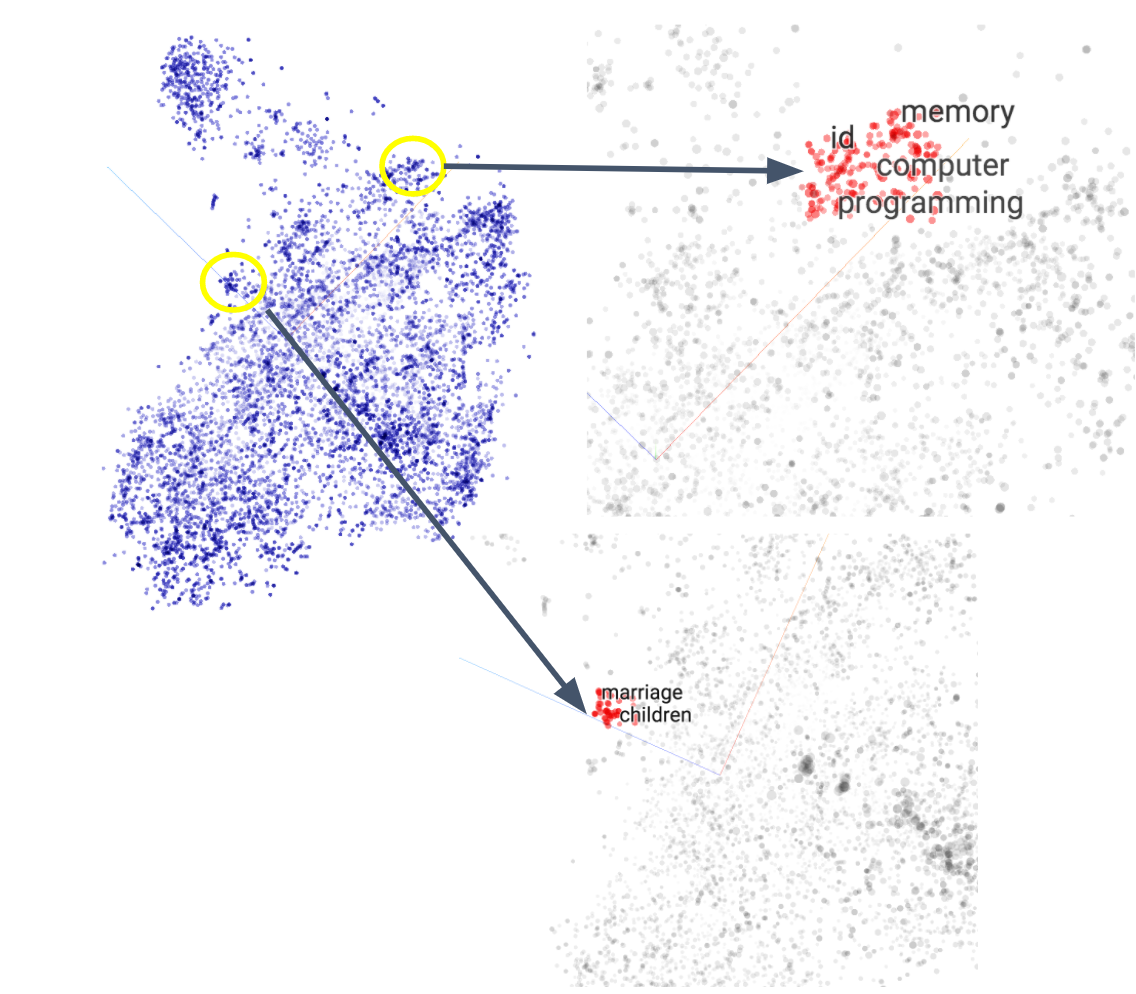


**Histograms:**

The change of Activations, biases and weights of 3 kinds of filters



**Embedding Visualization:**

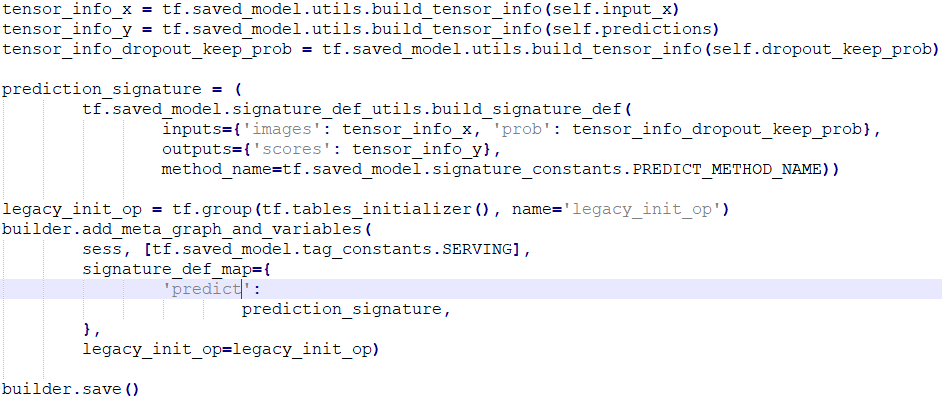


**4. Publish Web Service**

We used TensorFlow Serving components to export a trained TensorFlow Convolutional Neural Network (CNN) model and used the standard tensorflow\_model\_server to serve it.

**4.1 Train and Export TensorFlow Model**

We used SavedModelBuilder inside Tensorflow to export the trained model to a local directory. The builder basically saves the Session, input and output parameters as key information for future serving, as shown by Fig 4.1.1 below.



**Fig 4.1.1**

**4.2 Load Exported Model with Standard TensorFlow ModelServer**

First ModelServer package needs to be installed.

To add TensorFlow Serving distribution URI as a package source:

echo "deb [arch=amd64] http://storage.googleapis.com/tensorflow-serving-apt stable tensorflow-model-server tensorflow-model-server-universal" | sudo tee /etc/apt/sources.list.d/tensorflow-serving.list

curl https://storage.googleapis.com/tensorflow-serving-apt/tensorflow-serving.release.pub.gpg | sudo apt-key add -

To install and update TensorFlow ModelServer

sudo apt-get update && sudo apt-get install tensorflow-model-server

After ModelServer is installed, run this command:

tensorflow\_model\_server --port=9001 --model\_name=cnn\_tweet --model\_base\_path=/tmp/cnn\_tweet\_model/

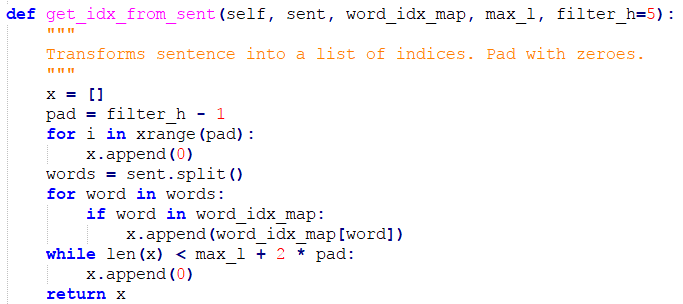
This command loads the exported model from the disk and exposes a port to serve the service.

**4.3 Test the Server**

First, to run Python client code, we can install the tensorflow-serving-api PIP package using:

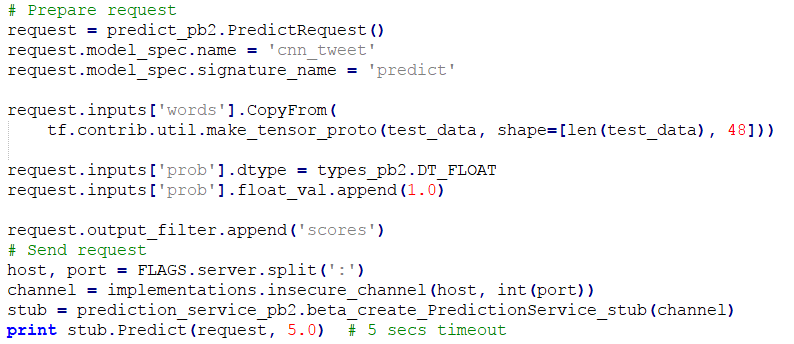
pip install tensorflow-serving-api

Then, we need to prepare the input data. We used Word2Vec package to spawn all the tweets and produce a word to index map, which was then used to get the index for each word, as shown in Fig 4.3.1 below. The index is a bridge that links a word and its vector representation.



**Fig 4.3.1**

Then we used the input data, e.g. word indices and drop probability, to send a request to the url on which the service is hosted, as illustrated in Fig 4.3.2 below.



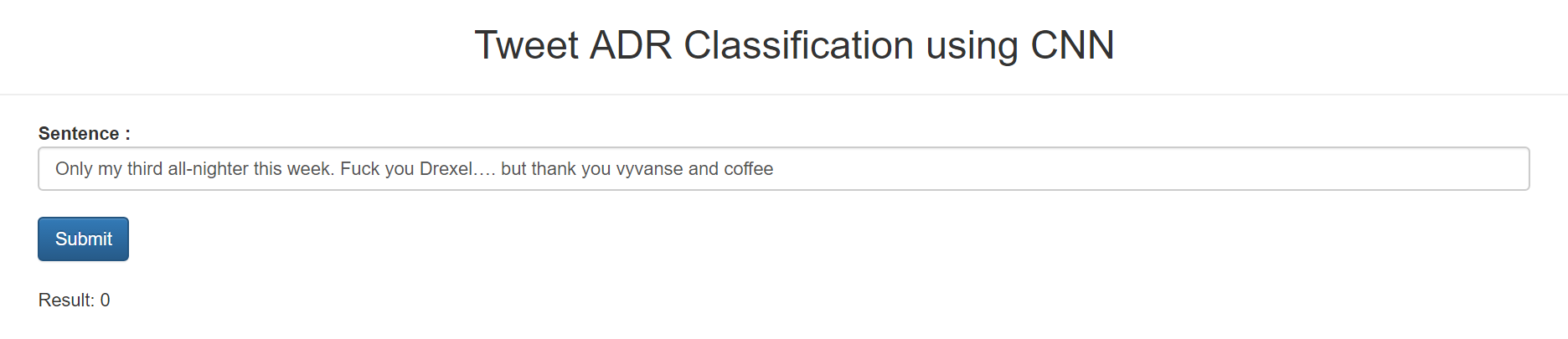
**Fig 4.3.2**

**5. Personas and User Interface Mockup**

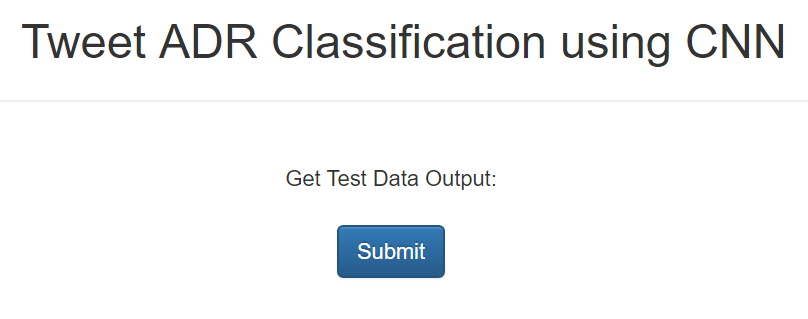
**Personas:**

Drug and medicine companies and employees will use our product to determine whether a tweet mentions ADR, so that they can better analyze the cause of the ADRs and improve their product.

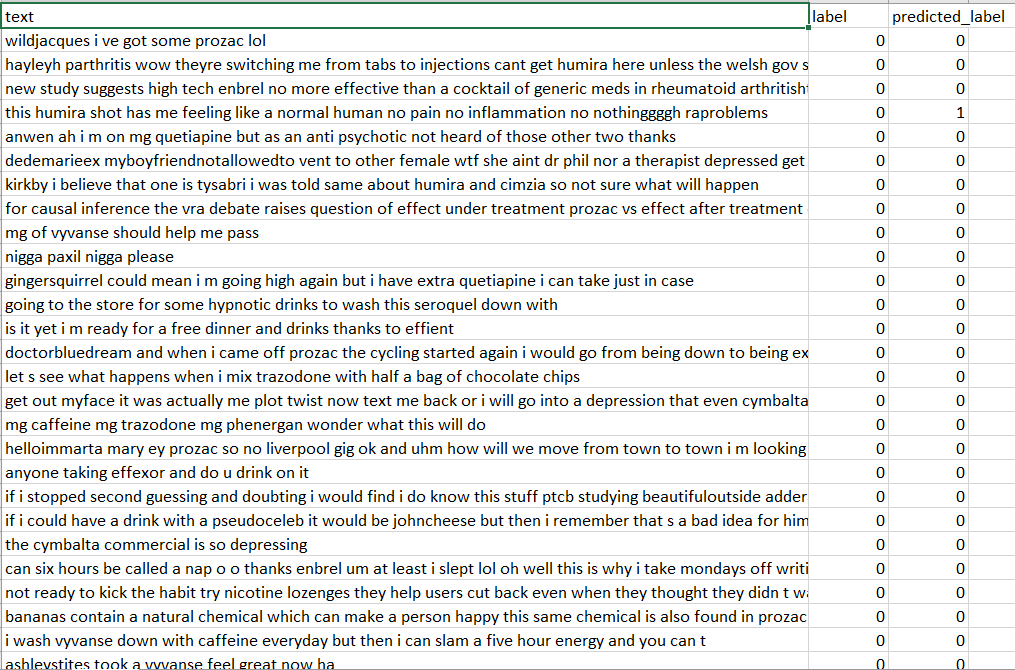
**User Interface Mockup:**

****

Please find the user interface mockup in the figure above. An user from a drug or medicine company can enter the Content of a tweet. And our system will use the classification model, which has been exposed as a service, to determine if that tweet contains any information about ADR. The result will be displayed at the bottom. 0 means ADR absence, while 1 means ADR presence.

****

Users can get the test data output by clicking on the button from the figure above. And a file will be downloaded like the figure below:



**6. Reference**

* [Convolutional Neural Networks for Sentence Classification](http://arxiv.org/abs/1408.5882)
* [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification](http://arxiv.org/abs/1510.03820)
* https://www.tensorflow.org/serving/serving\_basic