CSE P 517 - Final Project - Social Brand Experience

Apoorva Gupta (gapoorva@uw.edu)

March 19, 2021

1 Introduction

At my workplace, Qualtrics we offer software products and services for Experience Management. Experience Management refers to designing and improving experiences across a business to better the outcomes for all the people involved.

One of our core products at Qualtrics is Brand Experience, also known as ”BX”. With BX, we are offering a suite of software that is meant to democratize brand research, which has traditionally been a service that requires heavy investment in human capital and research experts. The major offering that we are building, and that this paper will focus on, is Brand Tracking, which refers to a time-series study that periodically collects brand experience data and visualizes how brands preform relative to each other in a particular market over time. Today, the product is primarily focused on vizualizing data pulled from surveys or questionnaires that are fielded to paid respondents. This is a standard industry practice, but it also presents some challenges.

Firstly, using a structured survey requires a respondent to be attentive for a period of time and respond to the survey truthfully. Often these surveys can be long, and in order to get a diverse sample size, we must pay third party companies to collect paid respondents. Because of this sampling method, we may not always get to hear from people who have had the most meaningful experiences in that particular market. In addition, because of this sampling technique, it's cost effective to run studies in periodic waves. These waves may be spread out over long periods of time from 2 weeks to almost quarterly. This increases the lead time between having an important event that significantly impacts a brand's standing within the market and having a measurement point that provides data on how that event may have reflected with consumer's experiences. Finally, this feedback mechanism may be too structured to collect freeform feedback from consumers. Often questionnaires are made of multiple choice questions and answers to reduce the burden upon the respondent. However, this sampling method is not completely lossless because a respondent's experience might not fit with the design of the question, and that meaningful feedback might be somewhat lost.

To address these points, I am proposing using social media data to extract much of the same information directly from consumers talking about the market or category of interest. To this end, we can build a new source of data that can be brought into the platform alongside survey data to measure brand experience in a fashion that is as real-time as possible.

In brand tracking, we are tracking a set of brands across a shared set of attributes. For example, one attribute we might choose is sentiment toward a brand, were we can develop a scaled score that represents average consumer sentiment. In this case, we are then looking to devlop a table that looks somewhat like the below. (Note that numbers are only examples).

Brand	**Sentiment at T=1**	**Sentiment at T=2**	**Sentiment at T=3**
Delta	0.7	0.74	0.64
United	0.2	0.31	0.49
Virgin Atlantic	0.93	0.91	0.95
Southwest	0.48	0.61	0.58
American	0.44	0.42	0.34

In this paper, I have chosen to focus on Twitter as a source of sentiment data. Twitter is used by 187 million people worldwide to share real-time reactions. There are also extensive data sets available in the public domain that were used to train a sentiment model. Twitter also provides a streaming API - in the future we could use this API to create a real time feed of sentiment data and stream it directly into our platform to drive educated insights and real action on consumers' on brand experiences. This helps brands better deliver on their brand promise, fix broken experiences, and improve outcomes for everyone involved.

2 Methods & Algorithms

2.1 Sentiment Classification

To get started with the sentiment scoring model, I first looked for a Twitter dataset that was tagged with sentiment scores. I choose to use the Sentiment140 Dataset which I found on Kaggle. This dataset included several fields per Tweet, but the relvant fields of interest were the "target" and "text" fields.

To prepare the data for input into the model, I used the "pandas" library to pull the CSV data and run several preprocessing steps on it. These were as follows:

- Recode the “target” label from a score from the discrete interval [0, 4] to a decimal between [0, 1]. This conforms to how we will build the model which will output a sigmoid activation.
- Remove special characters from tweets like ”@” symbols or hyperlink URLs.
- Optionally remove stopwords
- Run each word through NLTK’s Porter stemmer

After preprocessing of the tweets, we then use the keras Tokenizer to tokenize the documents and encode the target labels. We pad the document sequences to a length of 300.

We also use “genism” to build a Word2Vec embeddings model on our dataset. This is likely needed because natural language on social media platforms can be quite different from other sources. In this case, it will likely improve performance to create our own embeddings.

Finally, we define the model architecture and train it on the data! The model will use a pretrained word2vec embedding layer. In this study, I wanted to compare results using at least 2 different kinds of model architectures, one based on LSTM and CNN.

2.2 Sentiment Model Architectures

For the LSTM model, I used the architecture shown below:

```
self.model = Sequential()
self.model.add(embedding_layer)
self.model.add(Dropout(0.5))
self.model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
self.model.add(Dense(1, activation=’sigmoid’))
```

For the CNN model, I used the architecture shown below:

```
self.model = Sequential()
self.model.add(embedding_layer)
self.model.add(Conv1D(filters=32, kernel_size=8, activation=’relu’))
self.model.add(MaxPool1D(pool_size=2))
self.model.add(Flatten())
self.model.add(Dense(10, activation=’relu’))
self.model.add(Dense(1, activation=’sigmoid’))
```

2.3 NER

The second half of the project was to actually gain some brand experience insights by applying the model to another twitter dataset. For this part, I used an industry focused dataset, Twitter US Airline Sentiment, also found on Kaggle. From this dataset, we were mostly only interested in the text of tweets. To reduce the scope of the experiment, I choose to use an off-the-shelf NER tagger, spaCy, to tag organizations within tweets.

After tagging Tweets with relvant brands, I ran the sentiment classifier on text to produce a sentiment score. Then, averaged the scores to produce an average score per brand. Since the dataset was not tagged with timestamps, I assumed the entire dataset was part of one interval. In a business application, we might buffer collections of tweets from discrete time intervals and report average sentiment score by brand within such intervals.

3 Results

3.1 LSTM model

For the LSTM model, below was the training progress on the Sentiment140 dataset:

```
Train on 1280000 samples, validate on 320000 samples
Epoch 1/3
1280000/1280000 [==============================]
- 12330s 10ms/step
- loss: 0.4763
- accuracy: 0.7719
- val_loss: 0.5850
- val_accuracy: 0.7184
Epoch 2/3
1280000/1280000 [==============================]
- 11857s 9ms/step
- loss: 0.4295
- accuracy: 0.8015
- val_loss: 0.5615
- val_accuracy: 0.7352
Epoch 3/3
1280000/1280000 [==============================]
- 11600s 9ms/step
- loss: 0.4174
- accuracy: 0.8083
- val_loss: 0.5929
- val_accuracy: 0.7198
```

When evaluated on the test set, I found the following metrics:

```
loss 1.1500306129455566
accuracy 0.3755020201206207
```

3.2 CNN Model

For the CNN model, below was the training progress on the Sentiment140 dataset:

```
Train on 1277234 samples, validate on 319309 samples
Epoch 1/3
1277234/1277234 [==============================]
- 4002s 3ms/step
- loss: 0.4727
- accuracy: 0.7727
- val_loss: 0.6753
- val_accuracy: 0.6467
Epoch 2/3
1277234/1277234 [==============================]
- 3727s 3ms/step
- loss: 0.4296
- accuracy: 0.8003
- val_loss: 0.6586
- val_accuracy: 0.6566
Epoch 3/3
1277234/1277234 [==============================]
- 4031s 3ms/step
- loss: 0.4179
- accuracy: 0.8072
- val_loss: 0.6408
- val_accuracy: 0.6720
{
    ’val_loss’: [
        0.6752884550325983,
        0.6585956029997263,
        0.6407791040418346
    ],
    ’val_accuracy’: [
        0.6466526389122009,
        0.6565614938735962,
        0.6719666719436646
    ],
    ’loss’: [
        0.47267582757228527,
        0.4296040873941037,
            0.41791689156787637,
        ],
        'accuracy': [
            0.77271116,
            0.80030596,
            0.8071708
        ],
        'lr': [0.001, 0.001, 0.001]}
```

When evaluated on the test set, I found the following metrics:

```
    loss 1.0659139156341553
    accuracy 0.36947789788246155
```

3.3 Model comparison

Comparing the LSTM and CNN models, given the limited training period it does not seem that either architecture made a big difference in performance. Both LSTM and CNN models vastly overfit the training data and have much worse accuracy on the test set than on the training data. However, its notable that the CNN network trained in roughly half the time the LSTM network did, and inference was also faster for this model. Given performance is nearly identical, it's interesting to consider that the CNN model performs just as well and is much faster.

3.4 NER model

Using spaCy for the NER task was fairly easy and worked with almost no setup. However, the results were not useful and I ran out of time to improve upon the off-the-shelf model. Below is a (tiny) sample of the output when using basic spaCy NER to find the average sentiment per entity where each entity is labeled as an ORG across the Airlines dataset:

```
Panamerican Cross Country Cup 0.6703896522521973
@United Brothers 0.45850831270217896
Panamerican Cross Country Cup Championship 0.45850831270217896
PJs & 0.8370766639709473
FTW 0.34670600295066833
United Incompetence 0.18202507495880127
UA7985 0.8587554693222046
Florida Everglades 0.7515755891799927
UA938 0.4075564742088318
Terrific 0.042188167572021484
Newark # 0.43233898282051086
FC 0.46257901191711426
LUV 0.457063119326319
```

```
@SydneyAirport 0.149659663438797
Loyal 0.950650542974472
PIA 0.7529068887233734
```

As shown above, spaCy is not a well-tuned NER model for the task at hand. We would expect the model to only recognize Airline companies, if we were considering commericial airliners and the brands that compete in that category. Some entities were mis-tagged like ”Newark” while some don’t seem like entities at all and are just captializations like ”PJs”.

Given more time and resources, I would like to tune the NER model to specifically focus on labeling entities that represent our known brands only. This might require preparing a dataset specifically for that task. I believe the language used on Twitter also has a big impact and that this dataset would need to be reflective of language from twitter. The model I used for the experiment was trained on wikipedia pages. Unfortunately, I don’t have more interesting results to share.

4 Conclusion

As a result of this project I got hands-on experience learning how to build a sentiment model using some of the prevalant opensource libraries available for deep learning. Although I was not able to reach an immediately usable end product to achieve my original motivation of capturing social brand data, I made significant progress in understanding the components of building and evaluating a model. I also was able to explore the differences between and LSTM and CNN approach to sentiment classification.

With more time, I would like to expand on this project in the following ways: (1) Try using a Transformer model like BERT and tune it to the sentiment task. (2) Accquire a dataset and train an NER model for recognizing brand names in tweets (3) Train models for epochs on better hardware (I used a Mac for all my training). (4) Hook up a running inference engine to the twitter API and stream live sentiment data!

I also learned that NLP projects can be difficult to complete in short periods of time because of the long lead time with training and experimentation with different hyperparameters and model architectures. In this project I significantly struggled with time and balancing other priorities. In future projects, I need to factor this into my considerations and run experiments on subsets of the data.