Modelling search volumes as a dynamic system responding to external events

Stefan Sabev

Master of Informatics

School of Informatics
University of Edinburgh

2014

Abstract

It is well known that some events might spark people's interest to fly to different destinations. In particular news events or sports events can quite easily make people search for a specific destination - for example the Champions League Quarter final draw increased the number of flight searches from Glasgow to Spain 6 times. The main goal of this project is to collect Twitter data initially and possibly several other news sources and extract the available event data. Afterwards we'd want to split it into two groups - those who spark people's interest and those don't. This can later be used to detect when people want to go somewhere based on the news channels around the world.

Table of Contents

Chapter 1

Introduction

In recent times social media has been of great interest to everyone. Everyone is trying to benefit from the vastness of the data available - companies are paying for sentiment analysis, targeted ads, promoted trends and so on.

With a base of 190 million active monthly users, Twitter© has turned into quite a promising source of data. Many are trying harness the power of social media and use it to some purpose - to see how their product or company is performing or predict something about the future such as flu outbreaks, etc.

The aim of this dissertation is to explore the effect that social media, and more specifically Twitter, has over online travel. The aim is to show that using Twitter we can build a model that will be able to predict certain shifts in demand on some airline routes. In order to do this, we will employ a basic event detection mechanism, which will alert every time something odd happens for a particular destination and subsequently try to match that to a change of flight search profile.

Due to the fact that this is quite a niche area and the flight search datasets are not publicly available, there has been no previous research in this area. There has been a lot of work done of Topic Detection and Tracking (TDT) and First News Detection. Sasa Petrovic at the University of Edinburgh has done a lot of work and he's an excellent example of TDT (http://homepages.inf.ed.ac.uk/s0894589/petrovic-thesis.pdf)

Chapter 2

Data collection

The first and most important part of this project was to start collecting the correct data, which was to be used in building the model later. Twitter offers quite a comprehensive API with a lot of attributes, however in order to reduce the daily volume of data I had to take the most relevant ones for me.

The attributes chosen to collect are:

- Text - the text of the tweet is the most important one, perhaps. Quite a lot of information can be extracted from it alone
- Id - the tweet id. Useful if we want to screen scrape for any additional information or just to provide a tidy small dataset of ids.
- ID Str - String representation of the above.
- Source - What is used to post the tweet. The Twitter website is marked as "web".
- Coordinates - Representation of the geographical location of the Tweet as reported by the utility used to post the tweet.
- Entities - This include hashtags, user mentions and urls included in the tweet. Could be taken from the text, but it's nicer to have them ready.
- Retweet count - the number of times the tweet was retweeted. Useful for any future models.
- Favourited - Indicates whether the tweet was favourited by people - an analogy of this would be a Facebook like.
- Language - The language of the tweet. We are capturing English language tweets at the moment, but put in place for future expansion into the multi-language domain
- Filter level - indicates the level of filtering applied to the stream.
- Place - Shows where the tweet was tweeted from. Gives more detail than coordinates - country, city name, coordinates as well. Not necessarily available

for all tweets.

The first stage of the project is to look only at tweets in English coming from the UK. That will allows us to predict and model the flight search volumes in the UK based on the mentions in the Twitter Stream.

The second stage would be to develop the model even further and add multi-language and multi-country support and employ more sophisticated models such as ARIMA or Auto-regressive Vector Models.

By selecting those particular tweet attributes and using the Streaming API, I managed to reduce the daily volume of data from ~6GB of data down to ~3.5GB/day. The amount of data accumulated at the time of writing is ~380 GB. The collector has been running successfully from September 2013, however there are some holes of the data caused by network outages or the script interacting with the Twitter Streaming API crashing.

The data on flight search volumes is kindly provided by Skyscanner. In order to ensure that there are no concerns with confidentiality I have anonymised the data.

Chapter 3

Exploration and analysis of the data set

Since the problem we face here is completely new, I had to delve into the data and explore the data as much as possible in order to understand the data set and find out what we can use in order to do our predictive modelling.

I have tried several different approaches to filtering the data and exploring the relationship between the two variables. The ones worth mentioning are:

- Using hashtags which contain a country or city name and taking their counts.
- Taking every tweet that has a city/country name in its text in a conduction with a travel related term from the list.

Another important aspect of the project is to take care of the data. Because of some teething problems both with the Skyscanner data and the Twitter Streaming API, there are some holes in it, which I have chosen to remote by filling in data generated by the Last 4 Fridays forecasting method.

3.1 Hashtags

Hashtag is one of the most important constructs by Twitter. Here's an example tweet:

@FunnyChap: Something witty and very well said **#jokeoftheweek**

The structure of the tweets is:

1. The name of the user is denoted with a @ before the username.
2. The text itself.
3. The hashtag (a special Twitter construct) is denoted with a #.

You can filter out and explore twitter content based on hashtags and usually every major event has its own hashtag. The ones which are most mentioned appear in a special section called ”Trending”.

This option was considered, because if it had worked it would have been what we would call a ”quick win”. It doesn’t require much processing, since what you’d need is just take the ready made list of hashtags and store all the results into a in-memory dictionary, which you’d then split by city/country. That reduced the overall size of the relevant tweets to ~3 GB. What was even greater is that we didn’t really require the text information itself, which made the working dataset even smaller.

However the overall counts were quite small and the distribution of the values was very noisy, which means that fitting a model to this would’ve been quite hard if not impossible.

3.2 Occurrences paired with travel terms

The next slightly more expensive in computational terms option was to look at the actual tweet content and to count the number of times a city/country name appeared in conjunction with a one word form a list of travel-related words:

```
    terms =
{ ”airport”: ””,
”check-in”: ””,
”fly”: ””,
”land”: ””,
”landing”: ””,
”plane”: ””,
”take off”: ””,
”destination”: ””
... }
```

The list is modelled as a dictionary, because dictionary lookups are $O(1)$, so iterating over the words in the tweet and checking whether they appear in the city/country dictionary or the travel terms proved to be the most efficient combination.

That gives us about ~10 GB of tweets to work with. The reduced dataset is still much smaller in comparison to the full one. Processing the whole usually takes about a day, because of the extensive lookups in the city/country dictionary and the travel terms one.

3.3 Finding the correlation between those variables

After discovering that the volumes of twitter counts from the second method are not so noisy, I decided to proceed and explore the correlation between the two variables.

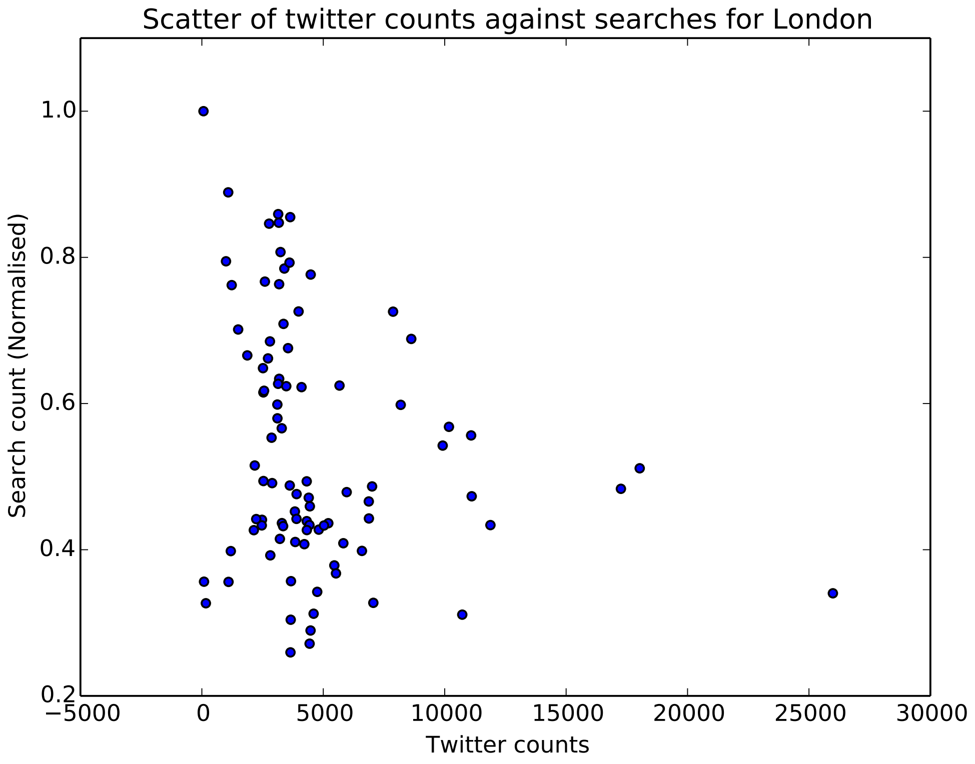
The easiest way to test the statistical correlation between your data would be to carry out the Pearson test.

For London the numbers we got back from that are:

```
In [39]: r_row, p_value
Out [39]: (0.13, 0.28)
```

From this it seems that the situation is truly unrelated. After all a positive correlation of only 13% is something that even a social scientist wouldn't report!

However, when you plot the two things we are trying to correlate something a bit more interesting comes up:



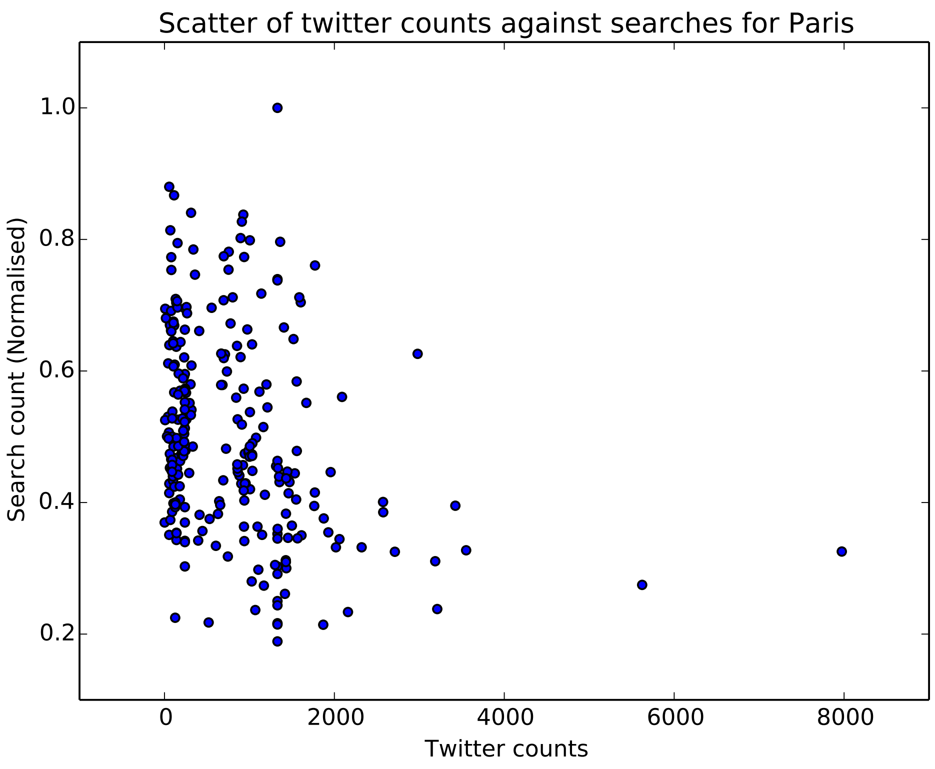
There is definitely a relationship, even though not perfectly linear.

We can observe that there is a well defined cluster where the twitter counts are big and the corresponding searches are relatively big as well. We can't say with great certainty that those two are perfectly correlated, because correlation does not imply causation. So what we needed to do was to delve even further and build a model that would work with those.

Paris was one of the most popular destinations so it was quite interesting to see what the relation is there. The Pearson test yields the following values for Paris:

```
In [24]: r_value, p_value
Out[24]: (-0.25, 0.0047)
```

And here is the scatter plot:



Here we do see that indeed the correlation is relatively small and negative.

As we can see each of the different cities shows a different behaviour. The implications of that are that the approach we take to building the model must be more versatile - we will aim to build one model for the overall numbers and models on a per city basis.

3.4 Cleansing the data

As mentioned, every time you depend a few external data source there will be some problems with the data, caused by several things.

In this project there were a couple of major sources of problems:

1. The script that collects data from the Streaming API.
2. The searches data coming from Skyscanner being incorrect or partial - not spanning the full date range.

The dates with incomplete data or missing data altogether can seriously impact any regression or statistical test, so it was vital to tidy up the data set by backfilling it. I applied the Last 4 Fridays forecasting algorithm mentioned in the following chapter in order to make the dataset more consistent and easier to work with.

The interesting part was that it completely change the results from the test. For London the values returned by the Pearson test are now:

```
In [23]: r_row, p_value
Out[23]: (-0.23, 0.02)
```

That is quite interesting and it has a lot of implications on the models we will use afterwards to correlate the two variables. We might have to add a certain ”lag” factor to this, which will offset the tweets by correlating the numbers extrapolated from Twitter with the numbers and match them against searches from the following day.

Chapter 4

Models

4.1 The baseline model

The baseline model for predicting the number of redirects Skyscanner has on a daily basis is called Last 4 Fridays. It works in the following way:

1. We want to predict the number of redirects/searches for this Friday.
2. We take the number of redirects/searches Friday from last week, the week before, etc, until we have the counts form the previous 4 weeks for the corresponding day.
3. We then assign weights to those 4 numbers and assign weights - the most recent one will get the highest weight, the one after about a half and so on. The exponential weighting scheme captures short term seasonality.

This model was tested out against more complex classifiers like ARIMA and Autoregressive Vector Methods and it performed really well, but unlike the others it was many times simpler to program and maintain.

4.2 Last 4 Fridays + Twitter Counts

The first model I build was not a radical new approach, but rather an increment on the previous one.

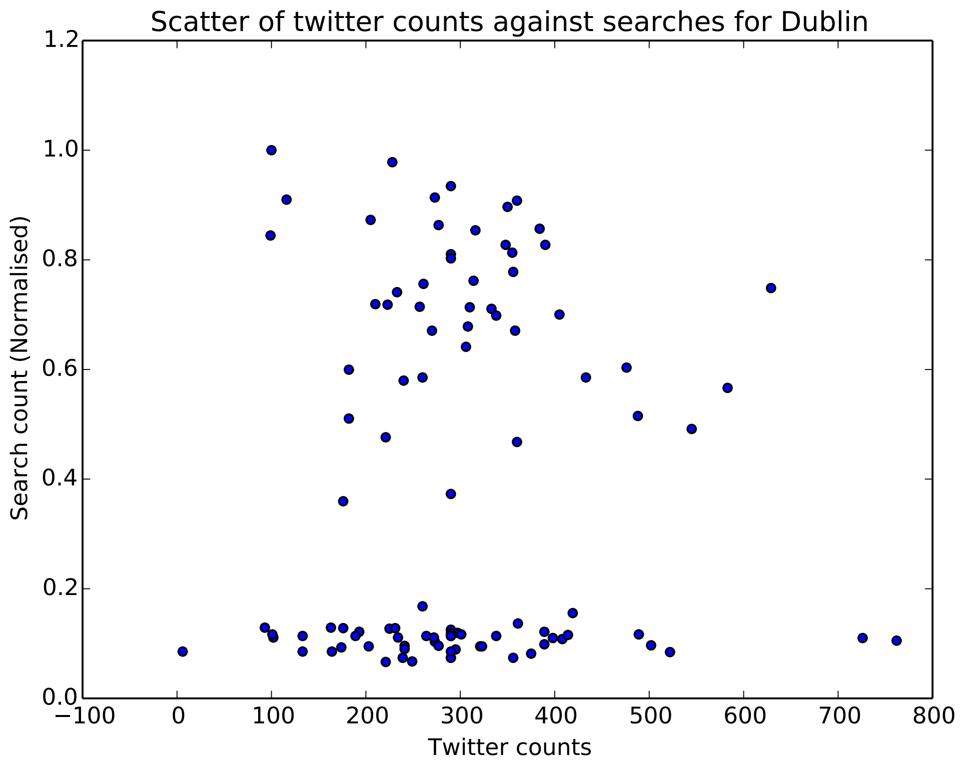
I generated a list of tuples which contained the following information:

- Twitter count for the place - how many times it was mentioned in conjunction with a travel term.
- Same day of the week from last week
- Same day of the week from 2 weeks ago
- Same day of the week from 3 weeks ago

- Same day of the week from 4 weeks ago

What this is is a version of Last 4 Fridays which now has Twitter Counts. Instead of using the exponential weights from the previous algorithm I let LASSO determine their weights. With LASSO I had weights for each of the 272 cities. After measuring the RMSEs for the L4F algorithm and expanded version with Twitter Counts the former performs better on 193 of the 272 cities. In the top 10 L4F+Twitter performs better on 2 out of 10.

Dublin which is in the 4th spot in terms of volumes is quite interesting. I plotted out the counts against searches just to see how it looks:



4.3 Results

And here are the results for the top 10 destinations by volume (they have the highest RMSEs).

City	RMSE L4F+Twitter	RMSE L4F
London	3335.869525	3160.05886
Paris	939.1057276	921.6785638
Barcelona	920.6831066	897.6473097
Milan	760.3436591	760.9286718
Rome	710.7052517	705.8388511
Manchester	574.8431422	572.4116201
Dublin	514.584231	527.884409
Amsterdam	550.2254781	516.2054596
Tenerife	544.6187529	502.0270723
Moscow	499.1803266	495.2139406

This is quite a good start considering the fact that all the weights were automatically determined by the LASSO algorithm and the fact that I have only about 130 data points so far (130 days).

4.4 Future improvements

The next on the list is to expand the feature set with more features from Twitter such as:

- Specific words.
- Trending event or not.
- Investigate whether sentiment will be useful here.

With those I'll be able to better the model and reduce the RMSE across the whole board and hopefully beat the current prediction algorithm for more than 80% of the examples.

Chapter 5

Timeline

The timeline for the next 3 months is the following:

1. Better the existing model and continue with the tests. .
2. Add more features to the set of inputs.
3. Work on both approaches:
    - Explain spikes in searches by looking at historical tweet data
    - Or try to predict the spikes in the first place
4. Final draft done by mid-March.
5. Submit dissertation the final week of March.