Tweets about 'scala', but not about 'scala'

Simon Hafner
{hafnersimon@gmail.com}

Abstract

Document relevancy is an important question that has been approached in various ways. With the advent of social media, especially Twitter, the documents of interest shrank in size. People tend to tweet a lot of information. The generated tweets are often only relevant to a very specific group of people and perceived as irrelevant by the rest. (Analytics, 2009) This phenomenon has been researched since the beginning of twitter. (Java et al., 2007) As a help to find informative tweets, the twitter bot allows a user to subscribe to a topic. To improve the accuracy of the tweet selection, relevant twitter users have to be specified. The bot then retweets only if the status is similar enough to what the specified users tweet. The bot itself also tweets about scala. If the bot tweets a tweet that is not relevant, it can be marked as irrelevant and this will be used to further improve the model. This idea is similar to NELL, (Carlson et al., 2010) and other projects of life-long learning. (Banko and Etzioni, 2007)

1 Introduction

Condensing streams of information can be a non-trivial task, as each user has a different perception of what renders a tweet relevant. This problem is approached with machine learning. Because machine learning results can be improved with response from the user, this program is implemented in a bot that can be responded to.

Twitter users are more interested in what is happening now than in outdated information, the relevancy decreases over time. (Teevan et al., 2011) This implies that the bot should retweet within a useful time frame, from seconds to minutes. This adds a real-time constraint to the solution. Training a model for another user should not interfere with the retweeting functionality of the bot.

A problem with using keywords as the basis for relevance evaluation is that words are ambiguous. The word 'scala' as an example has a wide range of possible meanings. [1]

This problem is approached by allowing the user to specify twitter users of interest. By analyzing the specified users, the bot knows which kind of 'scala' the user would like to hear about. This seeding is one option to achieve a disambiguation. Further methods that use a wider range of features are discussed in (Krishnamurthy and Mitchell, 2011). This dataset does not imply a problem with 'scala' disambiguation. [2]

The bot uses the keyword to ask twitter for tweets of relevance. A model is created from tweets from the users the relevant tweets should be similar to. Whenever a tweet is classified as relevant by this model, it is retweeted.

2 Detailed Description

The bot is built to satisfy three goals. It should always react to a new tweet within a reasonable time, preferably seconds. It should only retweet tweets that are relevant for the topic that is given by the user. It should learn from its mistakes, if it gets adequate feedback.

[1] http://en.wikipedia.org/wiki/Scala

[2] http://rtw.ml.cmu.edu/rtw/kbbrowser/programminglanguage:scala A search for 'scala' in the searchbox (I can't link it) reveals there are a lot of words containing 'scala', but nothing of relevance besides the programming language for only the word 'scala'.

2.1 Realtime

The bot must classify the tweet within a set time span, otherwise the bot does not stay up to date with the newest ideas on twitter. This is achieved by pre-training the model and training this model in the background. While the bot is functioning normally, using the old model. Whenever a user responds with a learning request, a new model has to be trained. This takes time, and the new model has to replace the old one as soon as it is ready.

2.2 Relevancy

The positive tweets for the model are taken from the last 2k tweets from each user that the interacting person specified. The maximum amount of tweets fetched this way is 3.2k.

After fetching the tweets, a twitter query for the keyword is issued and twitter users are collected from the answer. Any user that is followed by a specified user is dropped. For each user, the last 200 tweets are fetched. If it doesn't reduce the sample size too much, users with less than two or one mentions of the keyword are dropped.

Each tweet is tokenized with the ARK twitter POS tagger (Owoputi et al., 2013) and single Token, Tag, combined Token and Tag, as well as both Token bigrams and Tag bigrams are used as features. The keyword is removed from the tweet, as the negative examples may contain the keyword less often. This would lead to an imbalance of the model, as the tweets to be classified always contain the keyword.

Whenever twitter provides a tweet and it is classified to be a tweet relevant to the keyword, the model is asked to classify if it is also relevant given the relevant users. If the tweet achieves a score higher than 0.6, it is replied to with an '@$botuser' prepended, so it shows up in the tweet stream as relevant to that bot user.

The threshold is chosen low, because the bot can only be trained if it tweeted something wrong. This value might have to be implemented as adaptive to the current status of the rate limit.

2.3 Training the bot

The starting information is not much, so the first few classifications of tweets are uncertain. It is possible to tell the bot it shouldn't consider certain tweets as relevant in the future. Reply with 'no' or 'bad bot!' and it will learn from the experience by adding this tweet to the negative dataset twenty times.

This triggers a full retrain of the specified model and will take some time until it is active. Also, it has to fetch positive examples to balance the model. Those are currently taken from the same positive users, by fetching more tweets. This should lead to an improvement of the model.

2.4 Scala model

There is a model for the keyword 'scala' that is used by the bot itself so it retweets tweets that are relevant to the scala programming language. The positive tweets are created using a breadth-first search staring from 'etorreborre' and people that used the word 'scala' at least three times in their last 200 tweets.

The negative sample was taken from a query for 'scala' and the users were chosen when they used 'scala' at least two times in their last 200 tweets. Users that follow or are followed by 'etorreborre' were excluded. The users were also manually filtered to ensure the quality of the data. 2k tweets were collected per user to create this model. Whenever the keyword 'scala' is found in the stream from Twitter (as described above), it is checked for relevance.

The retweets are not true retweets, but rather a new tweet as a reply to the status to be retweeted with an RT beforehand. This is a technical limitation, as you cannot directly reply to a retweet. When replying to a retweet in the web interface, a mention is added to the original tweeter. When the user deletes that mention, twitter doesn't consider it a reply anymore. Replying to both original tweeter and bot would create noise for the original tweeter.

3 Sample interaction

A tweet such as shown below tells the bot to fetch the tweets as described above and starts to tweet as soon as it's ready.

```
@botty_anlp tweets about scala such as
```

```
etorreborre
```

The bot answers with a confirmation that it received your message and understood it.

```
@user Working on scala.
```

As soon as the models are trained, it will tell you it's ready.

```
@user Ready to tweet about scala.
```

When the bot tweets something it shouldn't, you can reply to it with 'no'. To circumvent twitter's limitation that the same tweet can only be posted once, additional text can be added after 'no'.

```
... retweet ...
@botty_anlp no don't do that
@user Sorry, I'll not make
 that mistake again.
```

4 Evaluation

For the evaluation of the tweet classification, the following command is simulated.

```
tweets about scala like
 etorreborre
```

The evaluation can be run with:

```
sbt 'run-main
 tshrdlu.util.RetweetEvaluation'
```

This gave 2k positive examples. From the precompiled corpus of the bot, the first five negative users were taken. From each of these users, 400 tweets were collected for the negative samples. This doesn't fully correspond to the 10 users and 200 tweets each for the bot, but the approximation should be good enough.

The threshold for a decision was set to 0.6, not to 0.5 as a default *nak* [3] model would do. This is to restrict the bot to not spam people with unrelated tweets.

The testing data is fetched from a twitter query for 'scala' and consists of 420 positive and 596 negative samples. The data was manually annotated. The data is split into 10 buckets with equal distribution of the positive and negative examples.

4.1 Baseline

The baseline would be to retweet everything. Given the sample distribution, this would give a recall of 100% and an accuracy of roughly 40%.

[3] https://github.com/scalanlp/nak

4.2 Baseline with Model

For the baseline, the model is created by the data compiled above. This serves as a simple baseline to compare to the other models.

Precision	Recall	F-Measure
74.84	71.35	66.86
78.38	72.88	67.60
73.68	66.61	59.81
75.00	64.41	55.70
71.46	62.37	53.59
81.22	80.17	77.19
71.92	63.22	54.88
66.04	64.04	59.92
75.35	72.20	67.94
72.32	67.11	61.28

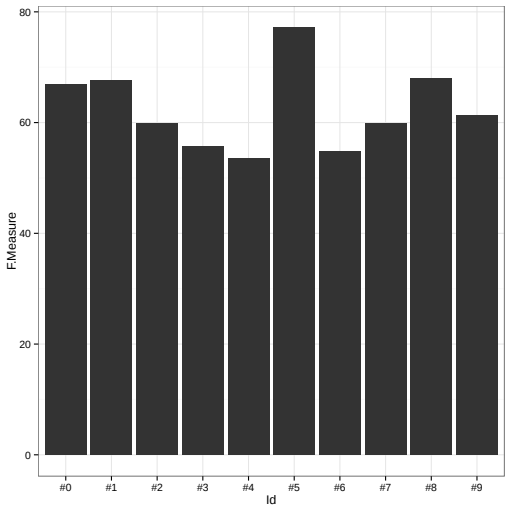

Figure 1: Baseline

4.3 Improving model

To simulate user feedback, 1/20 of the tweets were fed back into the model. As only tweets classified as positive will ever be retweeted, feedback can only be given to positive tweets. To increase training impact, each of those tweets was fed back twenty times. To balance the model, the same amount of positive tweets were added.

The order of exposure now plays a role. Depending on which tweets were learned, the model will give different answers. Running all possible permutations would be unfeasible, therefore the list of training sets is shifted after each run. This makes the last data set the first and moves all others one back. The results are averaged.

There is no significant improvement when training with only 5 tweets each. This might be because 5 tweets each is not enough to retrain the bot. The bot has a tendency to classify tweets as positive. Between 2/3 and 3/4 of the tweets are classified as positive.

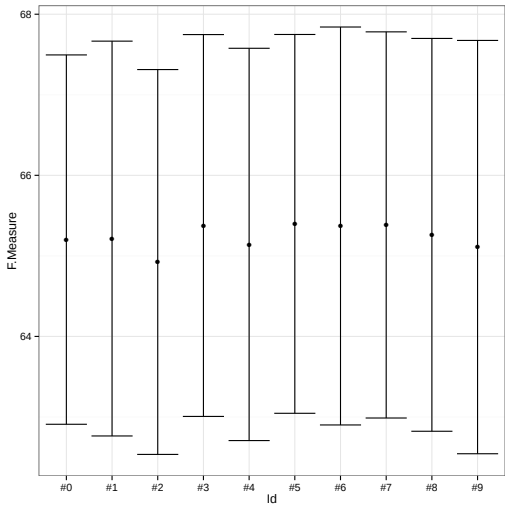

Figure 2: Grouped by position in the training sequence.

According to the data, there is not much difference in the different training sets. The order presented is not significantly relevant to the classification.

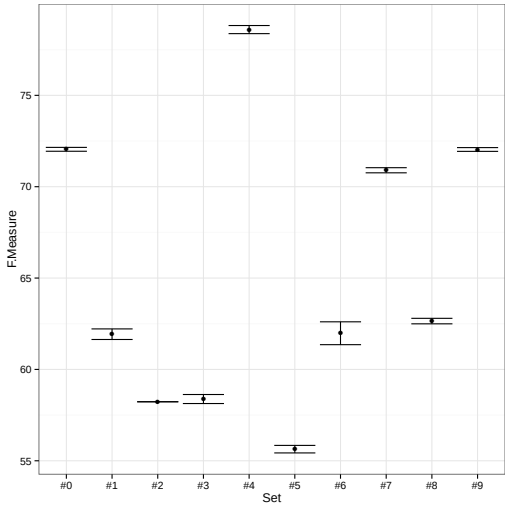

Figure 3: Grouped by bucket.

5 Future Work

So far, several issues have been encountered and not resolved. Some of them concern the implementation, others the NLP side of the project.

5.1 Using the follower graph

People have used the follower graph to recommend tweets. (Yan et al., 2012) This bot only makes marginal use of it. Using the rate limit as discussed below would make use of the follower graph, as additional information can be fetched from people that follow the specified users.

5.2 Quality over Quantity

A certain amount of quality control for the retweets would be of advantage. The quality depends heavyly on the users specified. This has been explored in previous work. (Huang et al., 2011)

5.3 Integration of external resources

URLs used in tweets are currently treated as text, without any added information. The referenced page could be fetched and indexed. There has been research which focuses on this part. (Duan et al., 2010) (Gao et al.,)

5.4 Better Assigning to the Models

Currently the bot checks for the existence of the keyword. However, some tweets, such as the following example, would be relevant but don't contain the keyword.

```
@clinton_freeman @markhibberd
 @mbrcknl I think @dibblego
 said: "understand FP: 5 mins,
 master FP: a lifetime" :-)
```

5.5 Language

The current model does not take language into account. Adding a language feature to the model would help to exclude tweets that are not of the same language as the specified users. Twitter provides this feature, but it is not implemented in twitter4j.

5.6 Rate Limit

Twitter limits the amount of queries possible, and the bot currently blocks or fails if the rate limit is hit. It would be of advantage to make the code aware of this limitation.

5.7 Training Access Control

There is no access control about training the bot. Anyone can tell the bot any tweet is considered bad and it will count each 'no' separately by adding negative examples to the dataset. This is only a yes/no question, so exposing it to the internet is no great risk. Exposing a natural language machine learning system to the internet is a risk. [4]

6 Conclusion

The bot is overly positive in its classifications. The experiment of a retweeting bot was a success, but the learning did not have the expected effect on the overall F-Measure. This might be because the training set was noisy.

7 Bibliography

References

Analytics, P. (2009). Twitter Study–August 2009. *San Antonio, TX: Pear Analytics. Available at: www. pearanalytics. com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009. pdf.*

Banko, M. and Etzioni, O. (2007). Strategies for lifelong knowledge extraction from the web. In *Proceedings of the 4th international conference on Knowledge capture*, pages 95–102.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., , E. R. H. J., and Mitchell, T. M. (2010). Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

Duan, Y., Jiang, L., Qin, T., Zhou, M., and Shum, H.-Y. (2010). An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 295–303.

Gao, D., Li, W., and Zhang, R. Beyond Twitter Text: A preliminary Study on Twitter Hyperlink and its Application.

Huang, M., Yang, Y., and Zhu, X. (2011). Quality-biased ranking of short texts in microblogging services. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 373–382.

Java, A., Song, X., Finin, T., and Tseng, B. (2007). Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA. ACM.

Krishnamurthy, J. and Mitchell, T. M. (2011). Which Noun Phrases Denote Which Concepts. In *Proceedings of the Forty Ninth Annual Meeting of the Association for Computational Linguistics*.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

Teevan, J., Ramage, D., and Morris, M. R. (2011). TwitterSearch: a comparison of microblog search and web search. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 35–44.

Yan, R., Lapata, M., and Li, X. (2012). Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 516–525.

[4] http://www.theatlantic.com/technology/archive/2013/01/ibms-watson-memorized-the-entire-urban-dictionary-then-his-overlords-had-to-delete-it/267047/