Article Topic Sentiment

Danish Shrestha
University of Illinois, Urbana-Champaign
dshrsth2@illinois.edu

Abstract

Simply performing a like/dislike for a news article sometimes lacks granularity. One has to either completely like or dislike the entire article. There is a need to perform like/dislike for an individual topic in an article. In this project we propose a way to extract celebrity names from an article and allow users to like/dislike an individual topic. We maintain a dictionary of celebrity and use lucene to match noun from an article against the dictionary. We also gather topic related text from an article and use a language based classifier to guess sentiment. The result were very promising if the topic in article was found in the dictionary.

Introduction

Like/dislike features has been very common in a lot of blogs, social media and news websites. Often times a news article talks about a bunch of different topics. A single like/dislike button isn't enough when you really want to only like a certain part of the article or a certain related context of an article. For example one might want to like the author and not necessarily the content of an article. Also, for a political campaign article, one might want to like one party and dislike the other party. Today, there isn't any tools out there that allows one to selectively like and dislike topics at a granular level.

In this project we crawled a bunch of articles from Thomson Reuters entertainment archive page and extracted topics from the doc. Topic was extracted from an article in two phases. In first phase, we ran the article through a POS tagger to filter out non noun terms. Then all the noun terms were fed into lucene so that we could match it against a crawled dictionary of topics. We limited topics to celebrity names and displayed them according to their relevance score. That way one could like dislike them at an individual basis. In addition to extracting topics, we also did some sentiment analysis to find the context for the specific topic. The sentiment analysis was done using a classifier to classify a topic between neutral, positive or negative.

The tool has an embedded web application that will help users open crawled articles and see results for themselves.

Related Work

One of the major task required to build an application that allows user to like/dislike more granular topics is to be able to automatically figure out topics related to the article. There are a lot of study related to topic extraction and name extraction from an article. Probabilistic topic modeling [1] can be very useful to extract topic without any supervision. The topics it generates may not be very closely related to each other but they would definitely be related to the articles.

In this project, we've focused more of extracting celebrities in entertainment news articles. We need a way to extract names from the article. The paper "Name Searching and Information Retrieval" mentions that names actually could be matched like any other terms and are found in any kinds of documents news, law, journals, etc [2]. Names specially can also be extracted from an article using a POS tagger. There has been a lot of research around this area. [3][4]. We've come far on tagging proper noun in a given article. This makes name extraction a lot more simple.

Problem Definition

The problem being solved in this paper is an inability for users to express their mixed opinion about an article without writing a comment. Its a lot easier for one to simply like/dislike an article. If there was a way to like/dislike an article in a more granular way and without writing a comment a lot of users opinion would be heard.

To solve this problem we simply need to be able to present topics for a given article and provide users with an ability to selectively like/dislike a topic. We also provide hint to the user by doing a sentiment analysis on the topic or sentences related to the topic. Computationally, it is no different than finding topics from an article and performing sentiment analysis on the topic. The major difference here is its application and how these techniques has been used for a very different use cases.

The input for the tool is news article that needs to be analyzed. The output is a list of topics sorted by their relevance and sentiment (neutral, positive, negative) for each topic.

Requirements

1. Ability to find topics and sentiments for the topics for a given article
2. Ability to like/dislike a topic

Methods

As discussed briefly in related work section above, we chose not to use latent topic model to find topics from an article. The main reason behind this is our focus was more in the celebrities. We were able to get better results by finding noun in the article and matching it against a large dictionary of celebrities. Also latent topic is very generic and doesn't provide much control over what the generated topics should be.

There are a couple of things we did here. First of all we built a crawler to crawl entertainment news articles sites like Thomson reuters entertainment articles [8]. In future we could add enhancement to crawl more sites. Or even better crawl the entire web and automatically determine if the article is an entertainment article or not and then act accordingly. Once the article has been crawled and saved to disk, we crawl pages like twitter 1000 famous peoples [9], posh24.com [10] to get a big list of hollywood celebrities.We stored both the articles and dictionary in the file system.

The dictionary is then added to lucene for indexing and searching. We would like to extract noun from an article and search against dictionary to see if the noun matches names in the dictionary. When a user opens up an article, we pass the article through a POS tagger to filter out non-noun words. Since all the celebrities names are noun, this works perfectly. We are now left with a handful of noun words. Some could be proper nouns like name of a person, celebrity and some could be common nouns. Matching it against the dictionary of name gave us a better confidence and helped us narrow down matches. We searched for all relevant names for each noun against the lucene index and stored the score in memory. We call this score a nounScore.

nounScore was very vague, we wanted a way to match the full name. So what we did next is got all retrieved documents (full names) from lucene. Added the entire article to lucene and now searched for each retrieved document against the article. We call this score a docScore. This would return all full names that matched in the article. This score was very reliable since it searched for a full name of a celebrity.

This was all great but despite us crawling celebrity names, there was quite a few names missing from the dictionary. This would not match any popular topic/names in the article but not present in the dictionary. To solve this problem we introduced another score derived via term weighting from words in article document. We call this score tfScore. Basically if there is a noun frequently occurring in the document it might be important. We really didn't care about inverse document frequency because we were not comparing multiple documents. As long as the noun was popular in a single document, it was good enough for our use case.

We basically now have three scores: nounScore, docScore and tfScore. The next challenge was how to combine them together. Logistic regression [6] was a good option but we ended up with simply adding weights in the form of 1-alpha * x + alpha * y. Here is the scoring formula we came up with. The structure of the document is very similar to

$$\text{score} = (1-\text{beta}) * (1-\text{alpha} * \text{nounScore} + \text{alpha} * \text{docScore}) + \text{beta} * \text{tfScore}$$

where

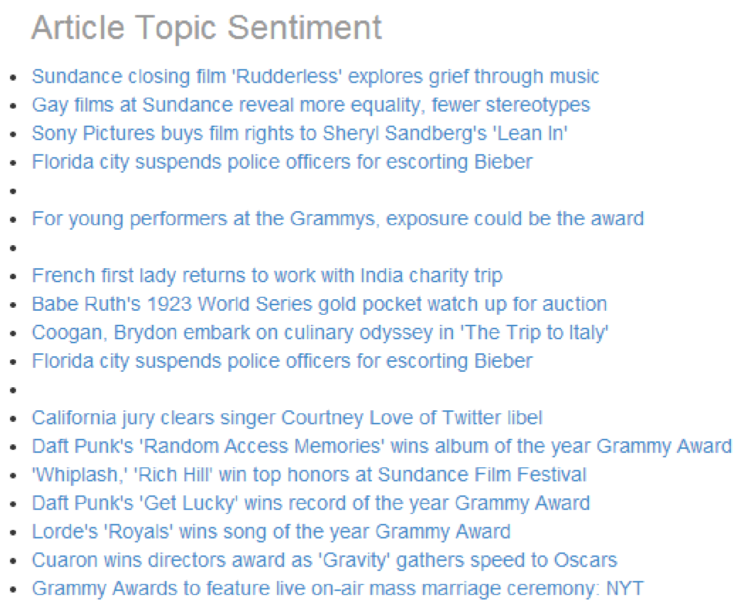
alpha is a parameter to weight nounScore or nounScore
beta is a parameter to weight tfScore

This still doesn’t solve all the problem. We don’t know what a good parameter value for alpha and beta would be. This is an optimization problem. To optimize the parameter values we ended up figuring out a way to evaluate the result. We will discuss more about evaluation below. Once we had a way to evaluate and compare against some evaluated results, we tuned different values and came up with the parameter value. We observed that docScore was actually very accurate and reliable since it was a direct dictionary match. We increasing the value of alpha to very close to 1 to give more weight to docScore.

Additionally, the system also does a sentiment judgement of text related to topic. Once we have a ranked set of topics, we searched it against lucene again. We leveraged the power of lucene’s hit highlighting to get text related to topic. Once we had a text we used a Lingpipe [5] classifier to guess if the text is neutral, positive or negative. Lingpipe uses a language model based classifier. It creates language model for positive, negative and neutral word distribution from user defined dataset. Once the language model is created, sentiments for the new documents are then classified. The classifier in this project is still very much experimental. We don’t have much evaluation around it. Our focus was mostly on extracting topics and evaluating topic extraction in this project.

Tool

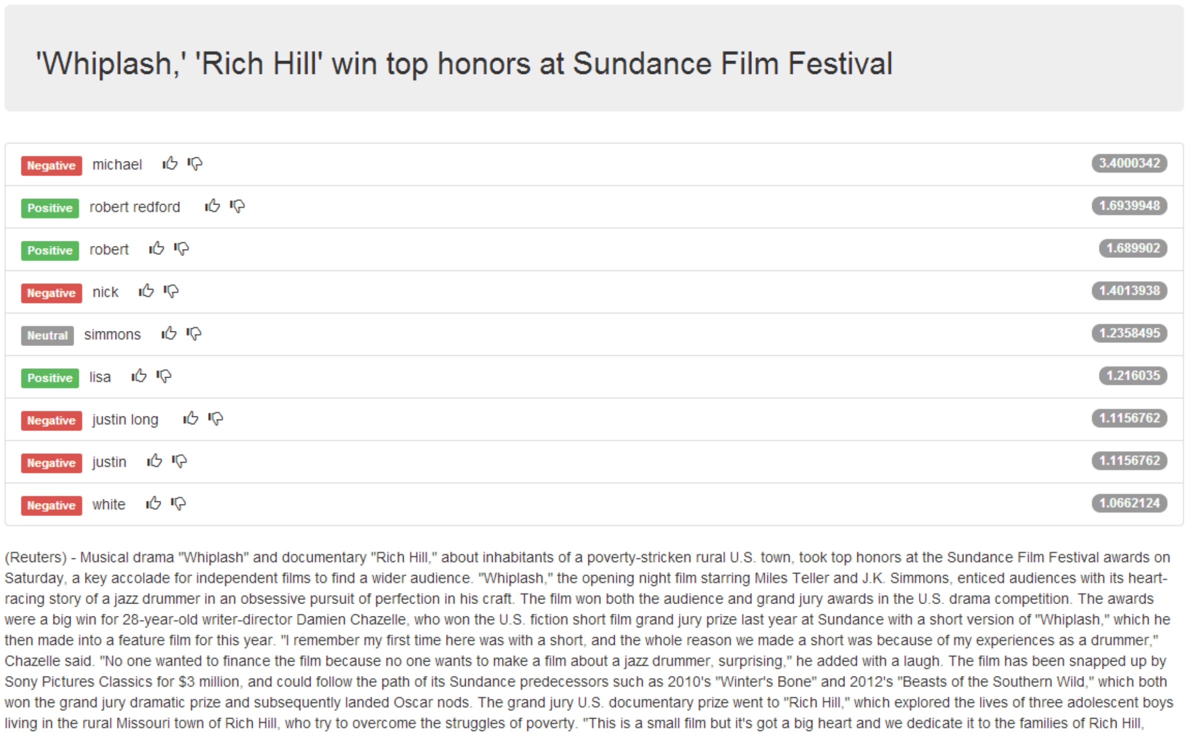
Here is a screenshot of the tool. The initial page (figure 1) simply displays list of all crawled articles. When one clicks on an article, it brings up the detail page. The detail page (figure contains the article itself and a rankec results. The ranked results in this case is all matched topics. The badge or the right is the score for each topic. The system also displays a sentimen for each matched topic. Finally, there is a like/dislike button next to eacl topic. This button will help user like dislike topics in a more fine-grainec manner.



Article Topic Sentiment

- Sundance closing film 'Rudderless' explores grief through music
- Gay films at Sundance reveal more equality, fewer stereotypes
- Sony Pictures buys film rights to Sheryl Sandberg's 'Lean In'
- Florida city suspends police officers for escorting Bieber
-
- For young performers at the Grammys, exposure could be the award
-
- French first lady returns to work with India charity trip
- Babe Ruth's 1923 World Series gold pocket watch up for auction
- Coogan, Brydon embark on culinary odyssey in 'The Trip to Italy'
- Florida city suspends police officers for escorting Bieber
-
- California jury clears singer Courtney Love of Twitter libel
- Daft Punk's 'Random Access Memories' wins album of the year Grammy Award
- 'Whiplash,' 'Rich Hill' win top honors at Sundance Film Festival
- Daft Punk's 'Get Lucky' wins record of the year Grammy Award
- Lorde's 'Royals' wins song of the year Grammy Award
- Cuaron wins directors award as 'Gravity' gathers speed to Oscars
- Grammy Awards to feature live on-air mass marriage ceremony: NYT

Figure 1: Articles list page


'Whiplash,' 'Rich Hill' win top honors at Sundance Film Festival

Negative	michael	3.4000342
Positive	robert redford	1.6939948
Positive	robert	1.689902
Negative	nick	1.4013938
Neutral	simmons	1.2358495
Positive	lisa	1.216035
Negative	justin long	1.1156762
Negative	justin	1.1156762
Negative	white	1.0662124

(Reuters) - Musical drama "Whiplash" and documentary "Rich Hill," about inhabitants of a poverty-stricken rural U.S. town, took top honors at the Sundance Film Festival awards on Saturday, a key accolade for independent films to find a wider audience. "Whiplash," the opening night film starring Miles Teller and J.K. Simmons, enticed audiences with its heart-racing story of a jazz drummer in an obsessive pursuit of perfection in his craft. The film won both the audience and grand jury awards in the U.S. drama competition. The awards were a big win for 28-year-old writer-director Damien Chazelle, who won the U.S. fiction short film grand jury prize last year at Sundance with a short version of "Whiplash," which he then made into a feature film for this year. "I remember my first time here was with a short, and the whole reason we made a short was because of my experiences as a drummer," Chazelle said. "No one wanted to finance the film because no one wants to make a film about a jazz drummer, surprising," he added with a laugh. The film has been snapped up by Sony Pictures Classics for $3 million, and could follow the path of its Sundance predecessors such as 2010's "Winter's Bone" and 2012's "Beasts of the Southern Wild," which both won the grand jury dramatic prize and subsequently landed Oscar nods. The grand jury U.S. documentary prize went to "Rich Hill," which explored the lives of three adolescent boys living in the rural Missouri town of Rich Hill, who try to overcome the struggles of poverty. "This is a small film but it's got a big heart and we dedicate it to the families of Rich Hill,

Figure 2: Article details page

Evaluation/Sample Results

Before evaluating we spent some time randomly picking 15 articles and extracting topics and sentiment related to the topic manually. The problem is very similar to search engine where a query is an article and relevant documents are returned topics. One of the main difference between this tool and the search engine is that total matched document in this tool is very limited. We observed that total retrieved documents were below 15 most of the times.

Mean arithmetic precision (MAP) score was perfect method to evaluate the software. MAP value measures for the entire retrieved results for all queries. For all 15 hand picked documents we calculated the arithmetic precision and then calculated the mean of it to get the MAP. The MAP value we got for 15 documents was 0.047. The map value seemed very low because our matching algorithm was very strict. We couldn't simply match an article by firstname or simply by last name. Both firstname and lastname of a topic had to match for the topic to be relevant. Also, out of 15 documents the system was able to retrieve at least one relevant topic for 10 documents. The system is functional but there are still some room for improvements.

We observed that the MAP score was very low when there was no matched name in the dictionary. This probably is a result of giving high weight for alpha. One of the improvement we could make is by crawling more celebrity names. Wikipedia has a huge collection of people biography and profile metadata. Crawling those data would definitely help us get better result. Ideally, it would be nice to use latent topic model to simply find topics. Since our focus was more on getting celebrity names it was difficult to use generic topic model. Perhaps a generic topic model with some prior knowledge might be a good experiment.

Conclusions & Future Work

The tool basically allows users to look at articles and provide more granular feed back without actually writing a comment by simply hitting like/dislike button for one or more matched topics. This data could further be used automatically to extract more knowledge.

As discussed above, it would be nice to use a generic topic model and remove the dictionary dependency. Dictionary dependency would be difficult to maintain going forward. Generic topic model would allow us to expand the tool to more than celebrities. We could then retrieve topics like common nouns, places, politician, events and many more.

Incorporating feedback is another missing piece in the project. Depending upon user action like/dislike we can automatically rerank topics and provide better results for future requests.

In general, there is a huge potential in this project and its application will help both the news provider and the users. There are still some more work for it to be used in real world.

Reference

[1] Blei, D. (n.d.). Probabilistic topic models. . Retrieved , from http://www.cs.princeton.edu/~blei/papers/Blei2012.pdf

[2] Thompson, Paul, and C. Dozier. "Name searching and information retrieval."Proceedings of Second Conference on Empirical Methods in Natural Language Processing. 1997.

[3] Ratnaparkhi, Adwait. "A maximum entropy model for part-of-speech tagging."Proceedings of the conference on empirical methods in natural language processing. Vol. 1. 1996.

[4] Goldwater, Sharon, and Tom Griffiths. "A fully Bayesian approach to unsupervised part-of-speech tagging." Annual meeting-association for computational linguistics. Vol. 45. No. 1. 2007.

[5] LingPipe. Retrieved, from http://alias-i.com/lingpipe/

[6] Logistic Regression. Retrieved, from http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf

[7] Apache Lucene, Retrieved from http://lucene.apache.org/

[8] Thomson Reuters Entertainment News. Retrieved, from http://www.reuters.com/news/archive/entertainmentNews

[9] Twitter counter. Retrieved, from http://twittercounter.com/pages/100

[10] Posh24.com Celebrities. Retrieved, from http://www.posh24.com/celebrities