Translating a Classifier

Patrick Martin

February 11, 2018

1 Overview

1.1 The Data

In order to work on translating a classifier, I need similar data in different languages. The first thing I'm going to use is reddit, starting with March 2017 (and adding others if I need more data), using langdetect (https://pypi.python.org/pypi/langdetect) on comments that have at least 15 unique words. I'm going to try to pull 10k each of Spanish, French, and Italian. Unfortunately this is either slow or sparse, and is taking a while. Hopefully it will finish by tomorrow or something.

Alternatively/additionally, if I can find different-language wikipediae, I can use page category as the classes.

1.1.1 LID data

We can make sure the LID worked reasonably well by checking the subreddits represented

Subreddit	Count	Subreddit	Count	Subreddit	Count
argentina	4,632	france	7,772	italy	7,862
mexico	1,840	Quebec	1,052	oknotizie	525
podemos[1]	1,622	montreal	197	ItalyInformatica	351
chile	592	ParisComments	116	italy_SS	327
vzla	280	French	44	italygames	86
Argaming	87	Lyon	35	perlediritaly	69
Spanish	80	FiascoQc	34	lisolachece	62
uruguay	75	SquaredCircle_FR	32	Romania	61
PuertoRico	50	effondrement	27	ItaliaPersonalFinance	54
Colombia	44	melenchon	23	ItalyMotori	40

1.1.2 Subreddit corpora

Using the data from the LID stuff, we can also just create the corpora by using all the posts from some subreddits. I propose[2]

[1] Spanish political party

[2] We'll fix this later

Spanish	French	Italian
argentina	france	italy
mexico	Quebec	
chile	montreal	
vzla	Lyon	
uruguay		
Colombia		

2 Classifier

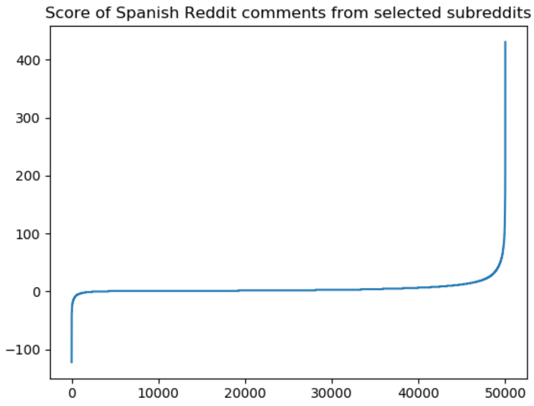
In order to make this work, we need to have some aspect that we're trying to classify against. One thing that has been done in the past is classification based on subreddit; based on the subreddits represented this could be possible: national subreddit (argentina, france, italy), gaming (Argaming, jeuxvideo, italygames), however maybe not much else. For reference, these are the fields I have to work with for each comment:

author	author_flair_css_class	author_flair_text	body	controversiality
created_utc	distinguished	edited	gilded	id
link_id	parent_id	retrieved_on	score	stickied
subreddit	subreddit_id			

Here are what I think:

- Controversiality
  Pro: Should be fairly language independent
  Con: Pretty sparse, only about 3% of the documents are "controversial". Controversial comments might tend to be in English, as well (comments from outsiders)

- Score: (binned)
  Pro: Should be fairly language independent
  Con: Borderlines between bins might be difficult, also more extreme scores are very sparse (¡ 0.2% have scores of at least 100)



- Gilded: (binary)
  Pro: Should be fairly language independent
  Con: Extremely sparse (¡ 0.03% have been gilded)

- Subreddit:
  Pro: Definitely influenced by context
  Con: Might be biased either towards games that are available in the given language, or the text will involve a lot of English.

- Flair (submissions):
  Pro: Human-decided topic labels *Con:* Submissions are not as common as comments, and the flairs across different subreddits might not align perfectly. Additionally, the title might not represent the content of the link perfectly.

The last idea just sort of came to me while looking at the subreddits. Using title text of submissions might be a good side project if this turns promising. To be specific, this is the number of comments in the smallest class for each of *gilded*, *controversial*, and *score*. (For *gilded* and *controversial*, this is the number of positive examples).

	Gilded	Contro	Score
Spanish-sub	3	347	123
Spanish	0	52	16
French-sub	1	423	70
French	0	64	10
Italian-sub	1	189	24
Italian	1	59	15

First, it's pretty clear that gilded isn't going to work well at all. Second, the bins for score might need to be refined.

3 Initial Classifiers

I divided my data into 80% train and 20% test and ran a couple classifiers on all of the data. Since *score* is multi-class, the default F1 score doesn't make sense, all of these classifiers will be evaluated on the harmonic mean of the fscores for each class[3]. I'm not sure if this is actually something that's really used, but it will work for our comparison purposes.

naive bayes			
	Gilded	Contro	Score
Spanish-sub	0.0100	0.0001	0.0002
Spanish	1.0000	0.0007	0.0013
French-sub	0.0392	0.0185	0.0002
French	err	0.0006	0.0004
Italian-sub	0.0392	0.0002	0.0004
Italian	0.0392	0.0006	0.0011

logistic regression			
	Gilded	Contro	Score
Spanish-sub	0.0100	0.1551	0.1145
Spanish	err	0.0004	0.1513
French-sub	0.0198	0.1889	0.0644
French	err	0.1523	0.0079
Italian-sub	0.0392	0.1902	0.0038
Italian	err	0.1198	0.0057

[3] Is this a real thing?

4 Bootstrapping

Recalling that controversial comments make up about 3% of the documents, and that classifiers are remarkably bad at picking them out, could we maybe train up a classifier to pick them out by bootstrapping? First, a look at the data:

	# Documents	# Controversial	%
Spanish	163,057	5,243	3.2%
French	207,348	9,449	4.6%
Italian	79,479	3,048	3.8%

We're going to look at Spanish first. Here is the (initial) algorithm:

1. Pick 5,000 documents at random (Random init)

2. Train some classifiers on those documents, splitting the 5,000 into random 80% train and 20% test sets (NOTE: these are not the same for each classifier) We'll use Naive Bayes (NB) and Logistic Regression (LR), run three different times. The table will show the one that works best

3. Pick the classifier with the best f-score, and run it over all the documents

4. Pick the 5,000 documents rated most likely to be controversial by that classifier, excluding any documents previously seen

5. Repeat

				Classifier f-score	
	Doc count	# Yes (repeats)	# Yes (total)	NB	LR
Random init	5000	172 (3.4%)	172	0	**0.087**
Round 1	5000	399 (8.0%)	437 (+265)	0	**0.077**
Round 2	5000	574 (11.5%)	683 (+246)	0	**0.16**
Round 3	5000	710 (14.2%)	928 (+245)	0	**0.097**

Some initial takeaways are that Naive Bayes just can't learn in this environment, but otherwise this process does work. Let's modify the algorithm to see if we can't make Naive Bayes a little happier:

1. Pick 5,000 documents at random (Random init)

2. Train some classifiers on *a subset of the documents, making sure positive documents comprise 33% of the dataset*, splitting *it* into random 80% train and 20% test sets (NOTE: these are not the same for each classifier) We'll use Naive Bayes (NB) and Logistic Regression (LR), run three different times. The table will show the one that works best

3. Pick the classifier with the best f-score, and run it over all the documents

4. Pick the 5,000 documents rated most likely to be controversial by that classifier, excluding any documents previously seen

5. Repeat

				Classifier f-score	
	Doc count	# Yes (repeats)	# Yes (total)	NB	LR
Random init	5000	173 (3.5%)	173	**0.16**	0.30
Round 1	5000	381 (7.6%)	429 (+256)	**0.36**	0.43
Round 2	5000	553 (11.1%)	702 (+273)	**0.31**	0.40
Round 3	5000	724 (14.5%)	1000 (+298)	**0.38**	0.47

Indeed, this did help Naive Bayes get on the scoreboard. Both it and LR improved drastically, although LR still remained ahead. For this batch I decided to bootstrap based on the best NB classifier, to see if it bootstrapped well. Indeed, the general results are very similar. How does this work for French?

Without subsetting					
				Classifier f-score	
	Doc count	# Yes (repeats)	# Yes (total)	NB	LR
Random init	5000	173 (3.5%)	173	**0.16**	0.30
Round 1	5000	381 (7.6%)	429 (+256)	**0.36**	0.43
Round 2	5000	553 (11.1%)	702 (+273)	**0.31**	0.40
Round 3	5000	724 (14.5%)	1000 (+298)	**0.38**	0.47

	D
Random init	50
Round 1	50
Round 2	50
Round 3	50