

Introduction and First Steps

The main aim of this assignment was to build a model using Apache spark *MLlib* which could predict labels (“safe”, “unsafe” and “vandal”) for the incoming Wikipedia edits stream. This report gives an overview of the main steps taken in order to achieve that and the final results. However, note that only some parts of code are displayed in this document. For the entire code with final outputs the reader should see files `spark_streaming_model.ipynb` and `spark_streaming_predicting_real_time.ipynb`.

After setting up Spark using the provided instructions and example notebooks, the dataset of Wikipedia edits was built. The edits were coming in every 10 seconds and saved in a separate folder starting with “myoutput” (or any other specified name) and followed by a unique string of numbers, as shown in Figure 1 (left). Each folder then contained an even number of files, usually between 2 and 14 (although sometimes there might be more depending on the number of incoming edits during the time interval). The files were given in two formats: `.crc` and file with no extension (Figure 1, right). The Wikipedia edit was saved in the latter file as a JSON dictionary. To indicate that the saving was carried out successfully, two files, `._SUCCESS.crc` and `._SUCCESS` (with no file extension) were always contained in the folder even if no edits were made during that time. The total number of collected edits was 58848.

myoutput-1586379960000	08/04/2020 11:06 PM	._SUCCESS.crc	08/04/2020 11:05 ...	CRC File	1 KB
myoutput-1586379970000	08/04/2020 11:06 PM	.part-00000.crc	08/04/2020 11:05 ...	CRC File	1 KB
myoutput-1586379990000	08/04/2020 11:06 PM	.part-00001.crc	08/04/2020 11:05 ...	CRC File	5 KB
myoutput-1586380000000	08/04/2020 11:06 PM	.part-00002.crc	08/04/2020 11:05 ...	CRC File	1 KB
myoutput-1586380010000	08/04/2020 11:06 PM	.part-00003.crc	08/04/2020 11:05 ...	CRC File	1 KB
myoutput-1586380020000	08/04/2020 11:07 PM	.part-00004.crc	08/04/2020 11:05 ...	CRC File	1 KB
myoutput-1586380030000	08/04/2020 11:07 PM	._SUCCESS	08/04/2020 11:05 ...	File	0 KB
myoutput-1586380040000	08/04/2020 11:07 PM	part-00000	08/04/2020 11:05 ...	File	18 KB
myoutput-1586380050000	08/04/2020 11:07 PM	part-00001	08/04/2020 11:05 ...	File	626 KB
myoutput-1586380060000	08/04/2020 11:07 PM	part-00002	08/04/2020 11:05 ...	File	18 KB
		part-00003	08/04/2020 11:05 ...	File	10 KB
		part-00004	08/04/2020 11:05 ...	File	104 KB

Figure 1. Saving Spark Streaming Output

All edits were put into one JSON data frame by looping through the subfolders containing them and using the *spark.read.json* function. A small part of the data frame where each row represents one instance (edit) from Wikipedia is shown in Figure 2 together with the code used to get it.

<pre> # LOOP THROUGH ALL FILES import os rootdir = 'C:/Users/Aistuxxe/Documents/spark/myoutput' files_list = [] for subdir, dirs, files in os.walk(rootdir): for name in files: if "part" in name.lower() and not ".crc" in name.lower(): files_list.append(os.path.join(subdir,name)) # PUT ALL FILES TO ONE JSON DATAFRAME df = spark.read.json(sc.textFile('.'.join(files_list))) df.show()</pre>						
comment	label	name_user	text_new	text_old	title_page	url_page
→References	safe	KingSkyLord	{{Infobox ice hoc...	{{Infobox ice hoc...	1959-60 Montreal ...	//en.wikipedia.or...
→top:missing tem...	safe	Pburka	{{Proposed deleti...	{{Proposed deleti...	Battle of Mpotona	//en.wikipedia.or...
(→Gameplay)	vandal	86.9.136.196	{{Use British Eng...	{{Use British Eng...	My Friend Pedro	//en.wikipedia.or...
(Added More.)	unsafe	AndersonL7333	{{Infobox politic...	{{Infobox politic...	Janata Dal	//en.wikipedia.or...
→Parodies and co...	safe	InedibleHulk	{{short descripti...	{{short descripti...	Smells Like Teen ...	//en.wikipedia.or...

Figure 2. JSON data frame and corresponding code.

Data Preprocessing

Before building a prediction model, we performed some data preprocessing. Firstly, we used the *ndiff* function from the *difflib* library to get the difference between the columns *text_old* and *text_new*. The code and example differences can be seen in Figure 3. We assumed that taking into account whether the text was added or deleted would not add much value to the prediction of classes. The reasoning behind it was that the deleted text in all three classes would be similar, since there would be no illogical / nonsensical words. Of course, one might hypothesize that the size is what matters in this case, i.e. “vandal” cases usually comprise large chunks of text being deleted. However, after exploring the data and in order to avoid overcomplicating the model, we assumed that plain difference is enough to distinguish between classes, regardless of whether the edit was made to add or delete text.

```
# DIFFERENCE BETWEEN OLD AND NEW TEXT
|
import difflib
import pyspark.sql.functions as F
from pyspark.sql.types import *

def make_diff(old, new):
    diff = difflib.ndiff(old, new)
    delta = ''.join(x[2:] for x in diff if x.startswith('- ') or x.startswith('+'))
    return delta

#convert to a UDF Function and get difference between columns
udfmake_diff = F.udf(make_diff, StringType())
df_difference = df.withColumn("difference", lit(udfmake_diff("text_old", "text_new")))
df_difference.show()
```

comment	label	name_user	text_new	text_old	title_page	url_page	difference
→References	safe	KingSkyLord	{{Infobox ice hoc...	{{Infobox ice hoc...	1959-60 Montreal ...	//en.wikipedia.or...	[[Category:1959 i...
→top:missing tem...	safe	Pburka	{{Proposed deleti...	{{Proposed deleti...	Battle of Mpotona	//en.wikipedia.or...	comment=
(→Gameplay)	vandal	86.9.136.196	{{Use British Eng...	{{Use British Eng...	My Friend Pedro	//en.wikipedia.or...	I LIKE COCAAAAAA...
(Added More.)	unsafe	AndersonL7333	{{Infobox politic...	{{Infobox politic...	Janata Dal	//en.wikipedia.or...]]
ideology	...						

Figure 3. Adding difference between *text_old* and *text_new*

After creating the *difference* column, we transformed it using tokenization (breaking text into words), normalization and stop word removal, using stop words provided by the *Ranks NL*¹. Then we created TF-IDF feature vectors by hashing the differences using the *HashingTF* function and rescaling them using *IDF*. Each difference was considered as a document and the whole collection of differences as the corpus. The reasoning behind this was that vandalism usually could be indicated by certain keywords, such as ‘mom’, ‘lol’, ‘Jews’, ‘follow’ (‘follow me on TikTok’) which were rare among edits but normal words in Wikipedia texts, while safe instances contained popular words, typo fixes, etc. Thus it made more sense to use *difference* and not, for example, *text_old* as the corpus. These TF-IDF vectors were later used as features for our Logistic Regression model.

Since the selected model accepted only numeric labels, we converted the categorical *label* column into a numeric one where the labels “safe”, “unsafe” and “vandal” corresponded to numeric labels 0, 1 and 2, respectively. Additionally, for convenience we put all of the above steps into a pipeline, trained it on the dataset (the training dataset or the whole dataset depending on whether it was model tuning stage or the final model fitting stage) and saved it to be used for predicting live Spark stream.

The final result of data preprocessing can be seen in Figure 4 where the *difference* column was obtained as explained above, *words* column corresponds to transformed *difference* after tokenization and normalization, *filtered* column is transformed *words* after stop words removal, *rawFeatures* and *features* columns are obtained using *HashingTF* and *IDF* functions,

¹ <https://www.ranks.nl/stopwords>

respectively, and *label* column is the numerical representative of the categorical *label* column (note, we renamed the categorical *label* column to *label_string* for convenience).

difference	words	filtered	rawFeatures	features	label
[[Category:1959 i... comment=	[category, 1959, ... [comment]]	[category, 1959, ... [comment]]	(10000,[3339,5347... (10000,[1294],[1.0])	(10000,[3339,5347... (10000,[1294],[2....	0.0 0.0
I LIKE COCAAAAAA...	[i, like, cocaaaa...	[like, cocaaaaaa...	(10000,[3330,4731... (10000,[3330,4731...	(10000,[3330,4731... (10000,[3330,4731...	2.0

Figure 4. Intermediate steps of featurizing the difference column

Finally, we also observed that the *comment* column had potential of being a good identifier of “safe” / “unsafe” / “vandal” classes, since the latter two cases often included nonsensical or out of the ordinary words or even no comments at all. Thus we preprocessed *comment* in the same way as the *difference* and combined both of them into one *features* column using the *VectorAssembler* function. In the modeling stage we used the features built from *difference* only and the combined *comment* and *difference* features separately, compared the results and then decided which features resulted in better accuracy (see section on Modeling).

Modeling

Before proceeding with the description of the modeling stage we have to note that using all 58848 collected edits to try, train and test the different models would have taken too long. None of our computers were able to handle such a large dataset without overheating and it would have taken over a week of non-stop code running only to select the best model not to mention rerunning the code on the whole dataset to fit the final model (considering that we also had to run codes for other assignments in parallel). Thus we selected a smaller subset of 1665 edits for the model selection and the final model training stages (label distribution can be seen in Table 2). The idea was to show our reasoning even if the final fitted model would not give extremely satisfying results.

Table 2. The number (percentage) of edits by label in dataset used for model selection

Safe	Unsafe	Vandal	Total
1405 (84.38%)	233 (13.99%)	27 (1.62%)	1665

In order to select and tune the model, we divided the featurized dataset from Table 2 to train and test sets (70% and 30%, respectively). The label distribution for both of them can be seen in Table 3.

Table 3. The number (percentage) of edits by label in training and testing datasets

Label	Safe	Unsafe	Vandal	Total
Training	995 (84.68%)	161 (13.70%)	19 (1.62%)	1175
Testing	410 (83.67%)	72 (14.69%)	8 (1.63%)	490

After some research, multiclass logistic regression was selected as the main model. Since the label classes were very unbalanced, we fitted both unweighted and weighted logistic regression with default parameter values, except for the maximum number of iterations (*maxIter*) which we set to 20 both because it was observed that this number was high enough for the model to converge and to limit the model training time. The corresponding weights used in weighted logistic regression were 1.18, 7.30 and 61.84 for safe, unsafe and vandal classes, respectively. The unweighted and weighted logistic regressions were fitted first using *features* based solely on difference and afterwards using *features* based on both difference and comments. The overall model accuracies and accuracies by label for train and test datasets can be seen in Tables 5 and 6. Accuracy measure here was chosen as a percentage of correctly classified edits (in the whole dataset / per label).

Table 5. Overall model accuracies and accuracies by label for the training dataset

	Unweighted Log. Regression		Weighted Log. Regression	
	Difference	Diff. + Comment	Difference	Diff. + Comment
Safe	99.50%	99.80%	99.00%	99.60%
Unsafe	92.02%	97.52%	95.71%	98.76%
Vandal	100.00%	100.00%	100.00%	100.00%
Overall	98.48%	99.49%	98.56%	99.49%

Table 6. Overall model accuracies and accuracies by label for the testing dataset

	Unweighted Log. Regression		Weighted Log. Regression	
	Difference	Diff. + Comment	Difference	Diff. + Comment
Safe	92.67%	90.73%	95.35%	92.93%
Unsafe	27.14%	31.94%	24.29%	33.33%
Vandal	0.00%	0.00%	0.00%	0.00%
Overall	81.22%	80.61%	83.35%	82.65%

Looking at the results for the training set (Table 5), we could see that all of the models learned vandal cases perfectly, which might imply overfitting (however, natural for training set), while the other two classes also had relatively high classification accuracies. However, weighted logistic regression with combined difference and comment features was noticeably better at predicting the other two labels as well. Table 6 indicated that this model was also the best at predicting unsafe cases in the test set although only second best at predicting safe cases. As expected, the underrepresented vandal class had 0% prediction accuracy for the test set, however we could argue that having higher accuracy in the unsafe class means that the model is able to pick up on unusual activity and with a larger training dataset the model would also learn to classify vandal cases better. Thus we selected weighted logistic regression with combined difference and comment features as the best model out of the four.

Since all of the above models were fitted using default parameter values (except for *maxIter*), the next modeling step would be to perform k-fold cross-validation on the selected model to tune its parameters such as *regParam* and *elasticNetParam*. The possible evaluation metrics for tuning parameters using the *MulticlassClassificationEvaluator* function include F1 measure, weighted precision, weighted recall and accuracy. Out of the four only the F1 measure is considered to be a satisfactory metric for imbalanced classification² (and it is the default one), while the latter three favour larger classes and neglect the infrequent ones³. The code for this step was written, however, the aforementioned limitations of the laptops in our possession prevented us from running it fully and getting the desired results. Thus the final model was selected to be simply the weighted logistic regression with combined difference and comment features. It was trained (together with the pipeline) on the whole dataset mentioned in Table 2 and saved for the real-time predicting of the incoming Spark stream of Wikipedia edits. Note that the pipeline was

² <https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/>

³ <https://kawahara.ca/weighted-precision-recall-equation/>

trained on the train set in such a way that the new incoming edits would be featurized using corpus from the train set.

Real-Time Predictions

After training and saving the final preprocessing and featurization pipeline, and logistic regression model, they were utilized to predict labels for the live Wikipedia edits data stream. Some of the live preprocessing and label prediction outputs are shown in the next two pages. Each output consists of three tables: the original Wikipedia edit that came in through the stream, the extracted features after preprocessing with the saved pipeline and the final prediction where *label* and *label_string* correspond to the true label, while the *prediction* and *prediction_string* correspond to the label predicted by the saved model.

As expected, the model was very good at predicting safe and unsafe cases. Of course, there were some cases when it predicted a safe edit instead of the unsafe one or vice versa. Regarding the vandal cases, the model usually predicted an unsafe label showing that it picked up on weird activity, but was too weak to detect vandalism. Tuning the model with cross-validation and training it on a larger dataset would possibly fix this issue.

```
ssc_t = StreamingThread(ssc)
ssc_t.start()
```

===== 2020-05-31 22:57:20 =====

comment	label	name_user	text_new	text_old	title_page	url_page
Adding localshort...	safe	Red Director	{{short descripti...	{{Infobox footbal...	Masaya Honda	//en.wikipedia.or...
→Descriptions in...	safe	Gre regiment	[[File:Fight Pygm...	[[File:Fight Pygm...	Pygmy (Greek myth...	///en.wikipedia.or...

difference	comment	features_diff	features_comment	features	label
{{short descripti...	Adding localshort...	(10000,[205,964,8...	(10000,[205,2626,...	(20000,[205,2626,...	0.0
[[[]]]	→Descriptions in...	(10000,[],[])	(10000,[2482,7586...	(20000,[2482,7586...	0.0

comment	difference	probability	label	label_string	prediction	prediction_string
Adding localshort...	{{short descripti...	[1.0,1.3932052190...	0.0	safe	0.0	safe
→Descriptions in...	[[[]]]	[0.54739470418626...	0.0	safe	0.0	safe

===== 2020-05-31 22:57:30 =====

comment	label	name_user	text_new	text_old	title_page	url_page
→Qualifications ...	safe	Pokeswap	The '''Chief Priv...	The '''Chief Priv...	Chief privacy off...	///en.wikipedia.or...
→Career	safe	Ytkr	{{Multiple issues...	{{Multiple issues...	Sudendu Shah	///en.wikipedia.or...

difference	comment	features_diff	features_comment	features	label
}}	→Qualifications ...	(10000,[],[])	(10000,[5396,6319...	(20000,[5396,6319...	0.0
he	→Career	(10000,[],[])	(10000,[2566],[4...	(20000,[2566],[4...	0.0

comment	difference	probability	label	label_string	prediction	prediction_string
→Qualifications ...	}}	[2.93897577671134...	0.0	safe	1.0	unsafe
→Career	he	[0.89724028386858...	0.0	safe	0.0	safe

===== 2020-05-31 22:59:10 =====

comment	label	name_user	text_new	text_old	title_page	url_page
→Can superflares...	safe	Bender235	{{about the extra...	{{about the extra...	Superflare	///en.wikipedia.or...
	unsafe	Invisible Devil	{{Use dmy dates d...	{{Use dmy dates d...	Shampa Reza	///en.wikipedia.or...

difference	comment	features_diff	features_comment	features	label
pdfabs.pdf	→Can superflares...	(10000,[789,4115]...	(10000,[20,427,95...	(20000,[20,427,95...	0.0
She is working a...		(10000,[521,1368,...	(10000,[],[])	(20000,[10521,113...	1.0

comment	difference	probability	label	label_string	prediction	prediction_string
→Can superflares...	pdfabs.pdf	[1.0,1.7906498683...	0.0	safe	0.0	safe
	She is working a...	[0.08762688090726...	1.0	unsafe	1.0	unsafe

===== 2020-05-31 23:04:40 =====

comment	label	name_user	text_new	text_old	title_page	url_page
ReplacedarXivPDF ...	safe	Bender235	In geometry, a ''... In geometry, a ''...	Substitution tiling	en.wikipedia.or...	
	unsafe	Sourav duir	{{short descripti... {{short descripti...	Amit Shah	en.wikipedia.or...	

difference	comment	features_diff	features_comment	features	label
pdfabspdfabs	ReplacedarXivPDF ...	(10000,[3837],[6....	(10000,[20,1709,2...	(20000,[20,1709,2...	0.0
20		(10000,[3746],[2....	(10000,[],[]	(20000,[13746],[2...	1.0

comment	difference	probability	label	label_string	prediction	prediction_string
ReplacedarXivPDF ...	pdfabspdfabs	[1.0,4.7480403239...	0.0	safe	0.0	safe
	20	[0.57808647162968...	1.0	unsafe	0.0	safe

===== 2020-05-31 23:07:30 =====

comment	label	name_user	text_new	text_old	title_page	url_page
→Response:Wikilink	safe	NedFausa	{{distinguish Bla... {{distinguish Bla...	Antifa (United St...	en.wikipedia.or...	
(→Death at sea) vandal	2601:98a:304:cb40...	{{other people J...	{{other people J...	John FitzAlan, 1s...	en.wikipedia.or...	
Adding localshort...	safe	Red Director	{{short descripti...	{{Infobox footbal...	Kohei Mihara (foo...	en.wikipedia.or...
→In Lithuania:m	safe	Ke an	{{About the Balti...	{{About the Balti...	Forest Brothers	en.wikipedia.or...

difference	comment	features_diff	features_comment	features	label
[[]]	→Response:Wikilink	(10000,[],[]	(10000,[1164,2445...	(20000,[1164,2445...	0.0
it said i could ...	(→Death at sea)	(10000,[1310,1850...	(10000,[5416,7275...	(20000,[5416,7275...	2.0
{{short descripti...	Adding localshort...	(10000,[205,964,8...	(10000,[205,2626,...	(20000,[205,2626,...	0.0
left	→In Lithuania:m	(10000,[1662],[5....	(10000,[6178,6638...	(20000,[6178,6638...	0.0

comment	difference	probability	label	label_string	prediction	prediction_string
→Response:Wikilink	[[]]	[0.99999997224121...	0.0	safe	0.0	safe
(→Death at sea)	it said i could ...	[3.94059564690059...	2.0	vandal	1.0	unsafe
Adding localshort...	{{short descripti...	[1.0,1.3932052190...	0.0	safe	0.0	safe
→In Lithuania:m	left	[0.99998163200471...	0.0	safe	0.0	safe

===== 2020-06-01 00:12:20 =====

comment	label	name_user	text_new	text_old	title_page	url_page
(→First-person s...	unsafe	190.178.207.227	{{short descripti...	{{short descripti...	Game engine	en.wikipedia.or...
	safe	Arkhandar	{{short descripti...	{{short descripti...	Rosalina (Mario)	en.wikipedia.or...
→Military	safe	RickMoretti	{{short descripti...	{{short descripti...	French name	en.wikipedia.or...
typo(s) fixed: Tr...	safe	Chris the speller	{{about the direc...	{{about the direc...	Mayor of Greater ...	en.wikipedia.or...
updated figure to...	safe	Scheng23	{{DISPLAYTITLE:''...	{{DISPLAYTITLE:''...	O-GlcNac	en.wikipedia.or...

difference	comment	features_diff	features_comment	features	label
RONALDO	(→First-person s...	(10000,[1069],[6....	(10000,[3183,5778...	(20000,[3183,5778...	1.0
Super 3DParty10World		(10000,[1340,2659...	(10000,[],[]	(20000,[11340,126...	0.0
Superiors address...	→Military	(10000,[126,398,8...	(10000,[5877],[5....	(20000,[5877,1012...	0.0
-_Express Transp...	typo(s) fixed: Tr...	(10000,[579,8962]...	(10000,[1139,1188...	(20000,[1139,1188...	0.0
bright alt= alt=					
updated figure to...		(10000,[1456,5196...	(10000,[2288,3587...	(20000,[2288,3587...	0.0

comment	difference	probability	label	label_string	prediction	prediction_string
(→First-person s...	RONALDO	[1.43211468765088...	1.0	unsafe	1.0	unsafe
	Super 3DParty10World	[0.96418490742483...	0.0	safe	0.0	safe
→Military	Superiors address...	[0.9999999984646...	0.0	safe	0.0	safe
typo(s) fixed: Tr...	-_Express Transp...	[1.0,7.1168850024...	0.0	safe	0.0	safe
updated figure to...	bright alt= alt=					
	[0.9999999738693...	0.0	safe			