NLP without Annotated Dataset

Building a Spam Classifier with Snorkel

Sowmya Vajjala

Seminar für Sprachwissenschaft, University of Tübingen, Germany

18 January 2021

Class Outline

- Generating training data for youtube spam classification
  1. Using labeling functions
  2. Using transformation functions
- Training a classifier, and seeing how it does
- Assignment 2 Description

Note: Today's class heavily relies on Snorkel's official spam classification tutorial. I did not upload my code yet - will do tonight or tomorrow after cleaning it up a little bit.
note: This is a long session staring at and discussing code snippets/screenshots. Take a deep breath!

Term papers schedule

- We are starting this Friday!
- Teams/Papers info on the forum.
- Presentation schedule (also posted on forum)
  1. 22nd January: Teams 1–3
  2. 25th January: Teams 4–7
  3. 27th January: Teams 7–9
- Format: 15-20 minutes of presentation, Around 10 min of discussion per team (everyone can ask questions. Presenters don't have to know all answers. I will also try to answer some of the questions that come up!)

Questions on this?

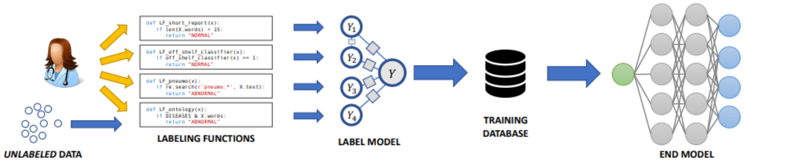
In the last class

- Anonymization of data in NLP
- Generating your own data
    - Weak supervision: An overview
    - Automatic Data Labeling: Some examples
    - Data Augmentation: Some examples
    - Text classification: An overview

-We will see how can data labeling and augmentation be used for text classification today. Any questions so far?

The Snorkel Pipeline

snorkel.org



```
def LF_short_report(x):
    if len(X.words) < 15:
        return "NORMAL"
```

```
def LF_off_shelf_classifier(x):
    if off_shelf_classifier(x) == 1:
        return "NORMAL"
```

```
def LF_pneumo(x):
    if re.search(r'pneumo.*', X.text):
        return "ABNORMAL"
```

```
def LF_ontology(x):
    if DISEASES & X.words:
        return "ABNORMAL"
```

Users write *labeling functions* to heuristically label data

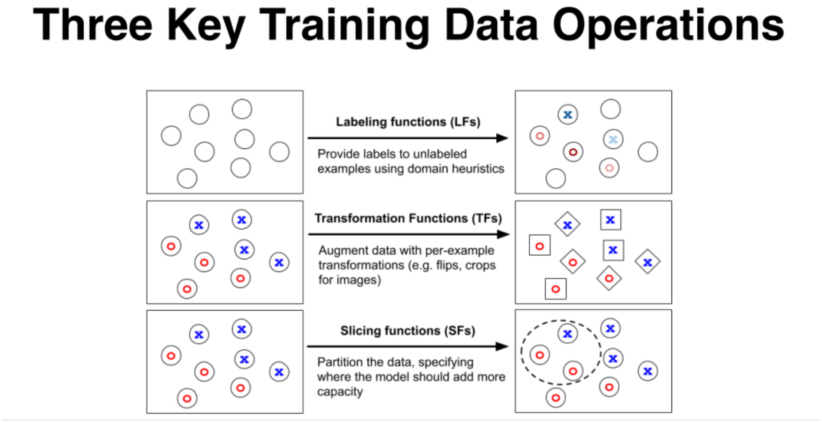
Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

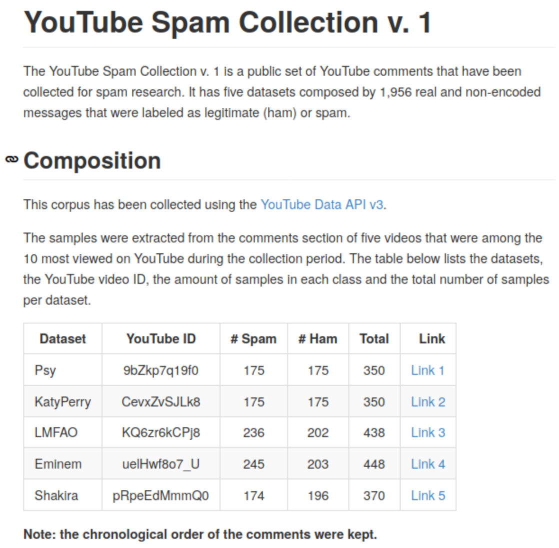
Note: No hand-labeled training data!

https://db.cs.washington.edu/events/workshop/2019/slides/alex-ratner.pdf

How to "create" data?


Three Key Training Data Operations
Labeling functions (LFs)
Provide labels to unlabeled examples using domain heuristics
Transformation Functions (TFs)
Augment data with per-example transformations (e.g. flips, crops for images)
Slicing functions (SFs)
Partition the data, specifying where the model should add more capacity

Source dataset



YouTube Spam Collection v. 1

The YouTube Spam Collection v. 1 is a public set of YouTube comments that have been collected for spam research. It has five datasets composed by 1,956 real and non-encoded messages that were labeled as legitimate (ham) or spam.

Composition

This corpus has been collected using the YouTube Data API v3.

The samples were extracted from the comments section of five videos that were among the 10 most viewed on YouTube during the collection period. The table below lists the datasets, the YouTube video ID, the amount of samples in each class and the total number of samples per dataset.

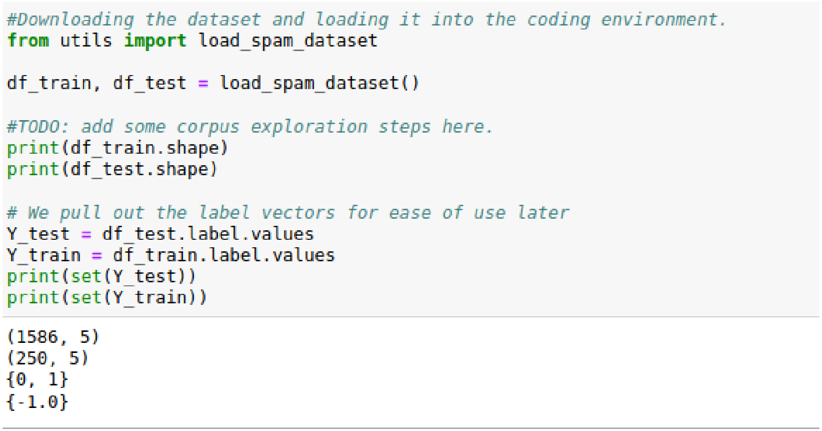
Dataset	YouTube ID	# Spam	# Ham	Total	Link
Psy	9bZkp7q19f0	175	175	350	Link 1
KatyPerry	CevxZvSJLk8	175	175	350	Link 2
LMFAO	KQ6zr6kCPj8	236	202	438	Link 3
Eminem	uelHwf8o7_U	245	203	448	Link 4
Shakira	pRpeEdMmmQ0	174	196	370	Link 5

Note: the chronological order of the comments were kept.

Source dataset

- The dataset contains comments from 5 of the most popular YouTube videos during a period between 2014 and 2015.
- The tutorial has some utility functions to load and explore their sample dataset. It has a load_spam_dataset() function that downloads the raw CSV files from the internet, divides them into splits, converts them into DataFrames, and shuffles them.
- The first four videos' comments are combined to form the train set. This set has no gold labels. The fifth video is part of the test set.

Loading the dataset



```
#Downloading the dataset and loading it into the coding environment.
from utils import load_spam_dataset

df_train, df_test = load_spam_dataset()

#TODO: add some corpus exploration steps here.
print(df_train.shape)
print(df_test.shape)

# We pull out the label vectors for ease of use later
Y_test = df_test.label.values
Y_train = df_train.label.values
print(set(Y_test))
print(set(Y_train))
```

```
(1586, 5)
(250, 5)
{0, 1}
{-1.0}
```

Note: The dataset has label information. However, we are here not going to use training set labels. We will use test set labels to evaluate our approach.

Labeling Functions

Labeling Functions

- LFs are heuristics that take as input a data point and either assign a label to it (in this case, ‘HAM‘ or ‘SPAM‘) or abstain (don’t assign any label).

Labeling Functions

- LFs are heuristics that take as input a data point and either assign a label to it (in this case, ‘HAM‘ or ‘SPAM‘) or abstain (don’t assign any label).
- They are a way of encoding some domain knowledge or some other relevant information to generate training data programmatically.

Labeling Functions

- LFs are heuristics that take as input a data point and either assign a label to it (in this case, ‘HAM‘ or ‘SPAM‘) or abstain (don’t assign any label).
- They are a way of encoding some domain knowledge or some other relevant information to generate training data programmatically.
- Labeling functions can be *noisy*: they don’t have perfect accuracy and don’t have to label every data point.
- Moreover, different labeling functions can overlap (label the same data point) and even conflict (assign different labels to the same data point).

They come in various forms

1. Keyword searches: looking for specific words in a sentence
2. Pattern matching: looking for specific syntactical patterns
3. Third-party models: using an pre-trained model (usually a model for a different task than the one at hand)
4. Distant supervision: using external knowledge base
5. Crowdworker labels: treating each crowdworker as a black-box function that assigns labels to subsets of the data

How do we develop LFs?

A typical LF development cycle consists of the following steps:

- Look at examples in training data to generate ideas for LFs
- Write an initial version of an LF
- Check how this dose in terms of its coverage, overlap with other LFs etc.
- Improve the LF or look for a new LF by inspecting the data further

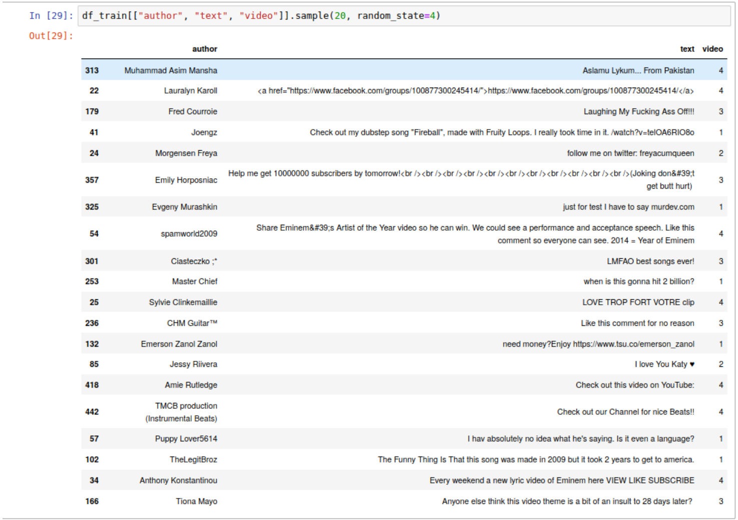
Things to keep in mind

- Our goal for LF development is to create a high quality set of training labels for our unlabeled dataset, not to label everything or directly create a model for inference using the LFs.

Things to keep in mind

- Our goal for LF development is to create a high quality set of training labels for our unlabeled dataset, not to label everything or directly create a model for inference using the LFs.
- The training labels are used to train a separate discriminative model (in this case, one which just uses the comment text) in order to generalize to new, unseen data points.
- Using this model, we can make predictions for data points that our LFs don't cover.

Exploring the dataset-1

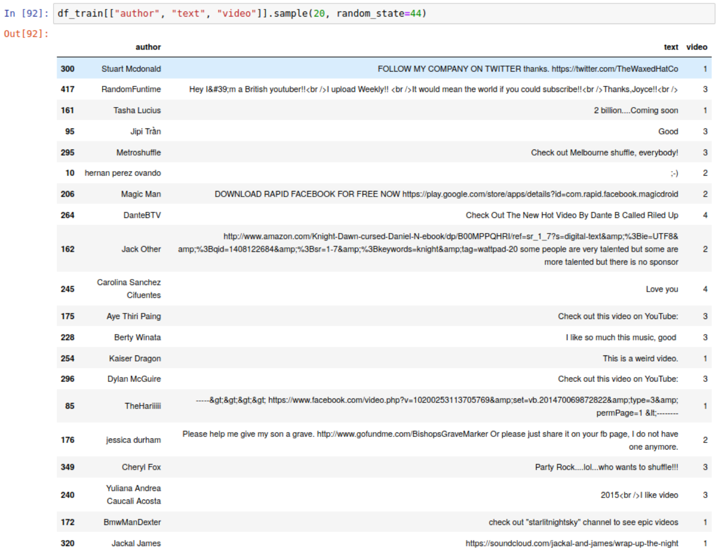


```
In [29]: df_train[["author", "text", "video"]].sample(20, random_state=4)
```

Out[29]:

	author	text	video
313	Muhammad Asim Mansha	Aslamu Lykum... From Pakistan	4
22	Lauralyn Karoll	<a href="https://www.facebook.com/groups/100877300245414/">https://www.facebook.com/groups/100877300245414/</a>	4
179	Fred Courroie	Laughing My Fucking Ass Off!!!	3
41	Joengz	Check out my dubstep song "Fireball", made with Fruity Loops. I really took time in it. /watch?v=telOA6RIO8o	1
24	Morgensen Freya	follow me on twitter: freyacumqueen	2
357	Emily Horposniac	Help me get 10000000 subscribers by tomorrow!<br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />(Joking don&#39;t get butt hurt)	3
325	Evgeny Murashkin	just for test I have to say murdev.com	1
54	spamworld2009	Share Eminem&#39;s Artist of the Year video so he can win. We could see a performance and acceptance speech. Like this comment so everyone can see. 2014 = Year of Eminem	4
301	Ciasteczko ;*	LMFAO best songs ever!	3
253	Master Chief	when is this gonna hit 2 billion?	1
25	Sylvie Clinkemaillie	LOVE TROP FORT VOTRE clip	4
236	CHM Guitar™	Like this comment for no reason	3
132	Emerson Zanol Zanol	need money?Enjoy https://www.tsu.co/emerson_zanol	1
85	Jessy Riivera	I love You Katy ♥	2
418	Amie Rutledge	Check out this video on YouTube:	4
442	TMCB production (Instrumental Beats)	Check out our Channel for nice Beats!!	4
57	Puppy Lover5614	I hav absolutely no idea what he's saying. Is it even a language?	1
102	TheLegitBroz	The Funny Thing Is That this song was made in 2009 but it took 2 years to get to america.	1
34	Anthony Konstantinou	Every weekend a new lyric video of Eminem here VIEW LIKE SUBSCRIBE	4
166	Tiona Mayo	Anyone else think this video theme is a bit of an insult to 28 days later?	3

Exploring the dataset -2



```
In [92]: df_train[["author", "text", "video"]].sample(20, random_state=44)
Out[92]:
```

	author	text	video
300	Stuart Mcdonald	FOLLOW MY COMPANY ON TWITTER thanks. https://twitter.com/TheWaxedHatCo	1
417	RandomFuntime	Hey I&#39;m a British youtuber!!<br />I upload Weekly!! <br />It would mean the world if you could subscribe!!<br />Thanks,Joyce!!<br />	3
161	Tasha Lucius	2 billion....Coming soon	1
95	Jipi Trần	Good	3
295	Metroshuffle	Check out Melbourne shuffle, everybody!	3
10	hernan perez ovando	;-)	2
206	Magic Man	DOWNLOAD RAPID FACEBOOK FOR FREE NOW https://play.google.com/store/apps/details?id=com.rapid.facebook.magicdroid	2
264	DanteBTV	Check Out The New Hot Video By Dante B Called Riled Up	4
162	Jack Other	http://www.amazon.com/Knight-Dawn-cursed-Daniel-N-ebook/dp/B00MPPQHRI/ref=sr_1_7?s=digital-text&amp;%3Bie=UTF8&amp;%3Bqid=1408122684&amp;%3Bsr=1-7&amp;%3Bkeywords=knight&amp;tag=wattpad-20 some people are very talented but some are more talented but there is no sponsor	2
245	Carolina Sanchez Cifuentes	Love you	4
175	Aye Thiri Paing	Check out this video on YouTube:	3
228	Berty Winata	I like so much this music, good	3
254	Kaiser Dragon	This is a weird video.	1
296	Dylan McGuire	Check out this video on YouTube:	3
85	TheHariiiii	-----&gt;&gt;&gt;&gt; https://www.facebook.com/video.php?v=10200253113705769&amp;set=vb.201470069872822&amp;type=3&amp;permPage=1 &lt;--------	1
176	jessica durham	Please help me give my son a grave. http://www.gofundme.com/BishopsGraveMarker Or please just share it on your fb page, I do not have one anymore.	2
349	Cheryl Fox	Party Rock....lol...who wants to shuffle!!!	3
240	Yuliana Andrea Caucali Acosta	2015<br />I like video	3
172	BmwManDexter	check out "starlitnightsky" channel to see epic videos	1
320	Jackal James	https://soundcloud.com/jackal-and-james/wrap-up-the-night	1

Before I start...

Let us do a small exercise. Under today's class (Topic 5), there is a folder with csv files - which is our dataset. In these csv files, there are comments and their classification, and some meta data. Ignore all other columns, just look at comments text (don't cheat and look at labels!) and see if you can identify what is spam and what is not, and if there are any patterns to that and if you can design some LFs (you dont have to code. Just plan). Look at the file with your group/room number.
Time: 20 minutes

Share your observations

Some Keyword LFs for Spam category

```
from snorkel.labeling import labeling_function
@labeling_function()
def subscribe(x):
    return SPAM if "subscribe" in x.text.lower() else ABSTAIN

@labeling_function()
def help_me(x):
    return SPAM if "help" in x.text.lower() else ABSTAIN

@labeling_function()
def need_money(x):
    return SPAM if "money" in x.text.lower() else ABSTAIN

@labeling_function()
def http(x):
    return SPAM if "http" in x.text.lower() else ABSTAIN
```

Some Keyword LFs for Ham category

```
@labeling_function()
def good(x):
    return HAM if "good" in x.text.lower() else ABSTAIN

@labeling_function()
def love(x):
    return HAM if "love" in x.text.lower() else ABSTAIN

@labeling_function()
def best(x):
    return HAM if "best" in x.text.lower() else ABSTAIN
```

Pattern based LF with regex

```
@labeling_function()
def check_out(x):
    return SPAM if re.search(r"check.*out", x.text, flags=re.I)
```

Heuristics based LF

```
@labeling_function()
def short_comment(x):
    """Ham comments are often short, such as 'cool video!'"""
    return HAM if len(x.text.split()) < 5 else ABSTAIN
```

LF with external tools

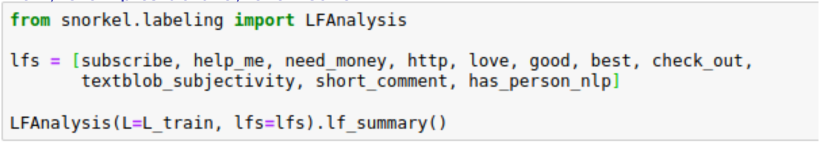
```
#memoize option is to cache the output so that functions
#that use this function don't have to rerun it each time.
@preprocessor(memoize=True)
def textblob_sentiment(x):
    scores = TextBlob(x.text)
    x.polarity = scores.sentiment.polarity
    x.subjectivity = scores.sentiment.subjectivity
    return x

@labeling_function(pre=[textblob_sentiment])
def textblob_subjectivity(x):
    return HAM if x.subjectivity >= 0.5 else ABSTAIN
```

LFs with more complex pre-processors

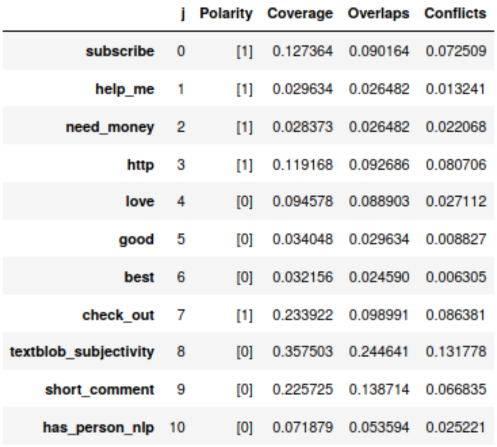
```
from snorkel.preprocess.nlp import SpacyPreprocessor

# The SpacyPreprocessor parses the text in text_field and
# stores the new enriched representation in doc_field
spacy = SpacyPreprocessor(text_field="text", doc_field="doc", memoize=True)

@labeling_function(pre=[spacy])
def has_person(x):
    """Ham comments mention specific people and are short."""
    if len(x.doc) < 20 and any([ent.label_ == "PERSON" for ent in x.doc.ents]):
        return HAM
    else:
        return ABSTAIN

from snorkel.labeling.lf.nlp import nlp_labeling_function

@nlp_labeling_function()
def has_person_nlp(x):
    """Ham comments mention specific people and are short."""
    if len(x.doc) < 20 and any([ent.label_ == "PERSON" for ent in x.doc.ents]):
        return HAM
    else:
        return ABSTAIN
```

Do these LFs do anything?

How many examples do they even cover?



```
from snorkel.labeling import LFAnalysis

lfs = [subscribe, help_me, need_money, http, love, good, best, check_out,
       textblob_subjectivity, short_comment, has_person_nlp]

LFAnalysis(L=L_train, lfs=lfs).lf_summary()
```



	j	Polarity	Coverage	Overlaps	Conflicts
subscribe	0	[1]	0.127364	0.090164	0.072509
help_me	1	[1]	0.029634	0.026482	0.013241
need_money	2	[1]	0.028373	0.026482	0.022068
http	3	[1]	0.119168	0.092686	0.080706
love	4	[0]	0.094578	0.088903	0.027112
good	5	[0]	0.034048	0.029634	0.008827
best	6	[0]	0.032156	0.024590	0.006305
check_out	7	[1]	0.233922	0.098991	0.086381
textblob_subjectivity	8	[0]	0.357503	0.244641	0.131778
short_comment	9	[0]	0.225725	0.138714	0.066835
has_person_nlp	10	[0]	0.071879	0.053594	0.025221

Coverage, Conflicts, FPs etc.

At this point, it is a good idea to check some sample labeled output for each LF, to ensure it is not too off-track.

- Coverage indicates how many examples in the training set does an LF cover.
- Conflicts indicate other LFs that reach a different categorization for a given LF.
- Overlaps indicate cases which are covered by at least one another LF.

We need good coverage, without too many conflicts, and without glaring false positives for an LF.

Applying LFs on the dataset

```
applier = PandasLFApplier(lfs=lfs)
L_train = applier.apply(df=df_train)
L_test = applier.apply(df=df_test)
```

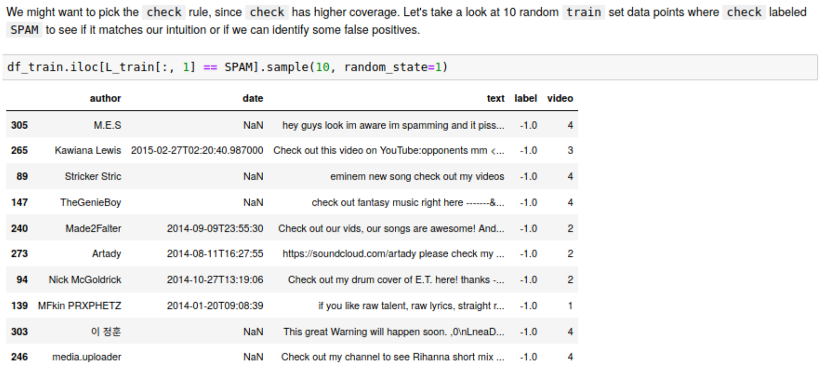
Since I have 11 LFs, each data point in train/test set is now a 11-dimensional vector, each dimension indicating an LF's decision for it.

Choosing among LFs

- Some of these LFs may just be overlapping a lot, or getting a lot of false positives or just too infrequent. How do we check for these?
- How do we compare the outputs of two LFs for a single (or a couple of data instances?)
- Snorkel has some built in tools for running this kind of analyses (note: I did not run these analyses!)
- There are also several other plotting tools (see the full snorkel tutorial for details) to do various analyses on LFs and their results.

LFs analysis - 1

How good is an LF?



We might want to pick the `check` rule, since `check` has higher coverage. Let's take a look at 10 random `train` set data points where `check` labeled `SPAM` to see if it matches our intuition or if we can identify some false positives.

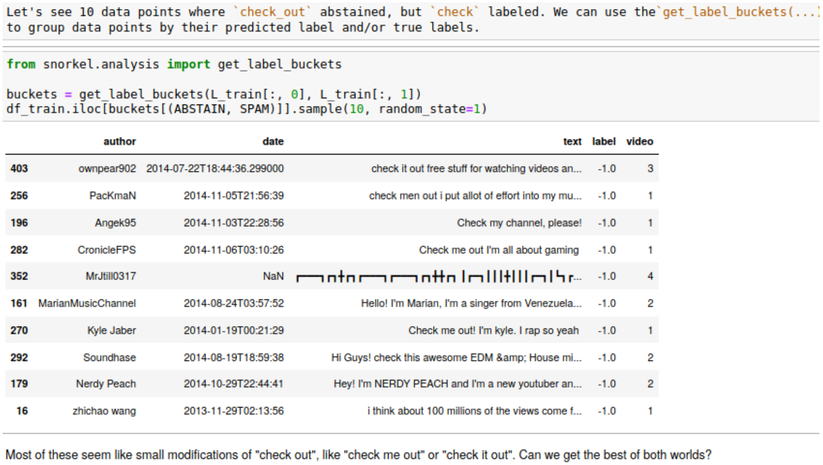
```
df_train.iloc[L_train[:, 1] == SPAM].sample(10, random_state=1)
```

	author	date	text	label	video
305	M.E.S	NaN	hey guys look im aware im spamming and it piss...	-1.0	4
265	Kawiana Lewis	2015-02-27T02:20:40.987000	Check out this video on YouTube:opponents mm <...	-1.0	3
89	Stricker Stric	NaN	eminem new song check out my videos	-1.0	4
147	TheGenieBoy	NaN	check out fantasy music right here -------&...	-1.0	4
240	Made2Falter	2014-09-09T23:55:30	Check out our vids, our songs are awesome! And...	-1.0	2
273	Artady	2014-08-11T16:27:55	https://soundcloud.com/artady please check my ...	-1.0	2
94	Nick McGoldrick	2014-10-27T13:19:06	Check out my drum cover of E.T. here! thanks -...	-1.0	2
139	MFkin PRXPHETZ	2014-01-20T09:08:39	if you like raw talent, raw lyrics, straight r...	-1.0	1
303	이 정훈	NaN	This great Warning will happen soon. ,0\nLneaD...	-1.0	4
246	media.uploader	NaN	Check out my channel to see Rihanna short mix ...	-1.0	4

note: "check" is the LF in second column here. Hence, the 1 in L_train[:1]

LFs analysis - 2

Comparing two LFs



Let's see 10 data points where `check_out` abstained, but `check` labeled. We can use the`get_label_buckets(...
to group data points by their predicted label and/or true labels.

```
from snorkel.analysis import get_label_buckets

buckets = get_label_buckets(L_train[:, 0], L_train[:, 1])
df_train.iloc[buckets[(ABSTAIN, SPAM)]].sample(10, random_state=1)
```

	author	date	text	label	video
403	ownpear902	2014-07-22T18:44:36.299000	check it out free stuff for watching videos an...	-1.0	3
256	PacKmaN	2014-11-05T21:56:39	check men out i put allot of effort into my mu...	-1.0	1
196	Angek95	2014-11-03T22:28:56	Check my channel, please!	-1.0	1
282	CronicleFPS	2014-11-06T03:10:26	Check me out I'm all about gaming	-1.0	1
352	MrJtill0317	NaN	[illegible]...	-1.0	4
161	MarianMusicChannel	2014-08-24T03:57:52	Hello! I'm Marian, I'm a singer from Venezuela...	-1.0	2
270	Kyle Jaber	2014-01-19T00:21:29	Check me out! I'm kyle. I rap so yeah	-1.0	1
292	Soundhase	2014-08-19T18:59:38	Hi Guys! check this awesome EDM &amp; House mi...	-1.0	2
179	Nerdy Peach	2014-10-29T22:44:41	Hey! I'm NERDY PEACH and I'm a new youtuber an...	-1.0	2
16	zhichao wang	2013-11-29T02:13:56	i think about 100 millions of the views come f...	-1.0	1

Most of these seem like small modifications of "check out", like "check me out" or "check it out". Can we get the best of both worlds?

note: here, check_out and check are first and second columns of L_train.

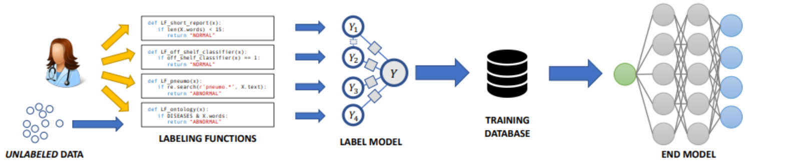
Okay, now what?

Once you do this analysis, you can just apply your final LFs list to generate a vector of labeled function outputs for each data point, using LFApplier from few slides before.

```
applier = PandasLFApplier(lfs=lfs)
L_train = applier.apply(df=df_train)
L_test = applier.apply(df=df_test)
```

The Snorkel Pipeline

snorkel.org



```
def LF_short_report(x):
    if len(X.words) < 15:
        return "NORMAL"
```

```
def LF_off_shelf_classifier(x):
    if off_shelf_classifier(x) == 1:
        return "NORMAL"
```

```
def LF_pneumo(x):
    if re.search(r'pneumo.*', X.text):
        return "ABNORMAL"
```

```
def LF_ontology(x):
    if DISEASES & X.words:
        return "ABNORMAL"
```

Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

Note: No hand-labeled training data!

28

https://db.cs.washington.edu/events/workshop/2019/slides/alex-ratner.pdf

From a vector of LF decisions to a "Label Model"

- Our goal is now to convert the labels from all our LFs into a single noise-aware probabilistic (or confidence-weighted) label per data point.

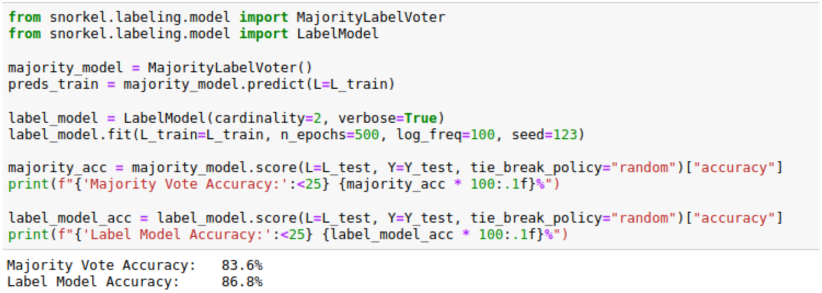
From a vector of LF decisions to a "Label Model"

- Our goal is now to convert the labels from all our LFs into a single noise-aware probabilistic (or confidence-weighted) label per data point.
- A simple baseline for doing this is to take the majority vote on a per-data point basis: if more LFs voted SPAM than HAM, label it SPAM (and vice versa). We can test this with the MajorityLabelVoter baseline model.

From a vector of LF decisions to a "Label Model"

- Our goal is now to convert the labels from all our LFs into a single noise-aware probabilistic (or confidence-weighted) label per data point.
- A simple baseline for doing this is to take the majority vote on a per-data point basis: if more LFs voted SPAM than HAM, label it SPAM (and vice versa). We can test this with the MajorityLabelVoter baseline model.
- However, not all LFs are born equal. They all have different coverages, accuracies etc.
- Snorkel has a sophisticated LabelModel, which learns to produce a single set of noise-aware training labels, which are probabilistic or confidence-weighted labels. (Algorithm paper)

Label Model

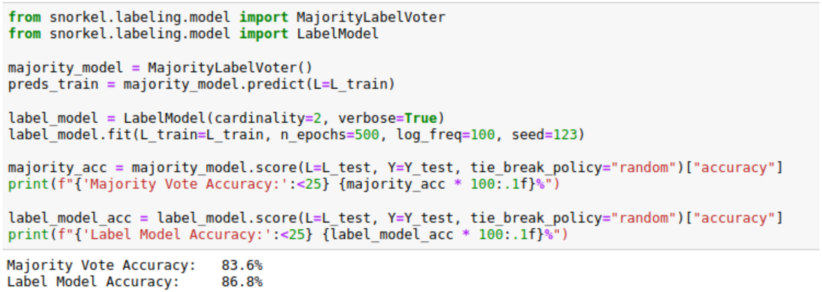


```
from snorkel.labeling.model import MajorityLabelVoter
from snorkel.labeling.model import LabelModel

majority_model = MajorityLabelVoter()
preds_train = majority_model.predict(L=L_train)

label_model = LabelModel(cardinality=2, verbose=True)
label_model.fit(L_train=L_train, n_epochs=500, log_freq=100, seed=123)

majority_acc = majority_model.score(L=L_test, Y=Y_test, tie_break_policy="random")["accuracy"]
print(f"{'Majority Vote Accuracy:':<25} {majority_acc * 100:.1f}%")

label_model_acc = label_model.score(L=L_test, Y=Y_test, tie_break_policy="random")["accuracy"]
print(f"{'Label Model Accuracy:':<25} {label_model_acc * 100:.1f}%")
```

```
Majority Vote Accuracy:   83.6%
Label Model Accuracy:     86.8%
```

Great. Are we done? Why? Why not?

Label Model



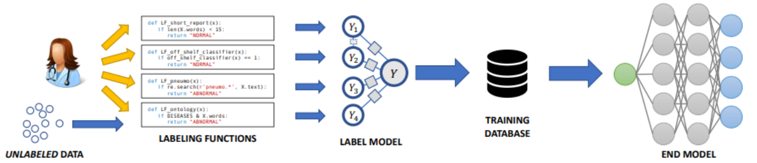
```
from snorkel.labeling.model import MajorityLabelVoter
from snorkel.labeling.model import LabelModel

majority_model = MajorityLabelVoter()
preds_train = majority_model.predict(L=L_train)

label_model = LabelModel(cardinality=2, verbose=True)
label_model.fit(L_train=L_train, n_epochs=500, log_freq=100, seed=123)

majority_acc = majority_model.score(L=L_test, Y=Y_test, tie_break_policy="random")["accuracy"]
print(f"{'Majority Vote Accuracy:':<25} {majority_acc * 100:.1f}%")

label_model_acc = label_model.score(L=L_test, Y=Y_test, tie_break_policy="random")["accuracy"]
print(f"{'Label Model Accuracy:':<25} {label_model_acc * 100:.1f}%")
```

```
Majority Vote Accuracy:   83.6%
Label Model Accuracy:     86.8%
```

Great. Are we done? Why? Why not?
We eventually want a model that does not need us to rely on LFs perpetually. We are also using LFs to only build this dataset. So, we use this Label Model as input to a "regular" classifier,

The Snorkel Pipeline

snorkel.org



```
def LF_short_report(x):
    if len(X.words) < 15:
        return "NORMAL"
```

```
def LF_off_shelf_classifier(x):
    if off_shelf_classifier(x) == 1:
        return "NORMAL"
```

```
def LF_pneumo(x):
    if re.search(r'pneumo.*', X.text):
        return "ABNORMAL"
```

```
def LF_ontology(x):
    if DISEASES & X.words:
        return "ABNORMAL"
```

Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

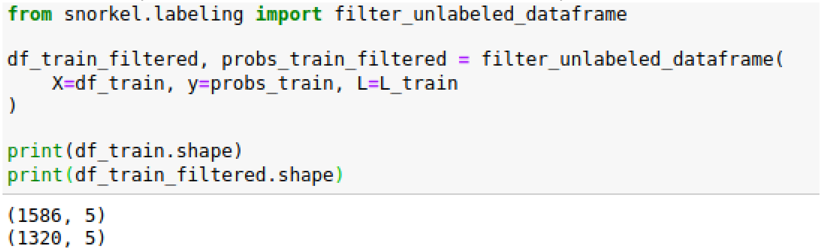
Note: No hand-labeled training data!

28

https://db.cs.washington.edu/events/workshop/2019/slides/alex-ratner.pdf

Filter out unlabeled datapoints

Since we wrote our LFs only based on observation, there may be some data points just not covered by any LF. These are not useful for "learning". They can be removed with snorkel's built in functions: (or we can add LFs to cover these!)



```
from snorkel.labeling import filter_unlabeled_dataframe

df_train_filtered, probs_train_filtered = filter_unlabeled_dataframe(
    X=df_train, y=probs_train, L=L_train
)

print(df_train.shape)
print(df_train_filtered.shape)
```

```
(1586, 5)
(1320, 5)
```

Looks like we lost about 250/1600 data points because of not concentrating much on LFs.

All set to train again!

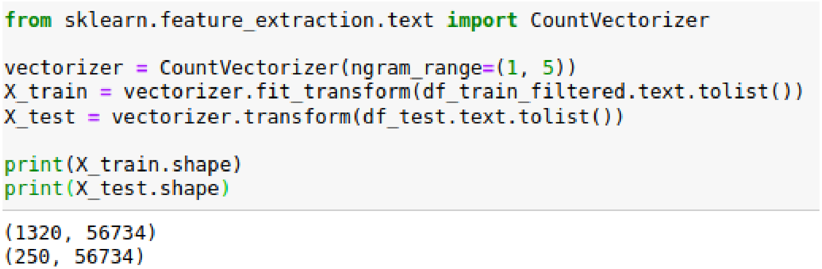
- Now, we have our "labeled" training data. Let us see how we do. How can we know?

All set to train again!

- Now, we have our "labeled" training data. Let us see how we do. How can we know?
- Remember that we have a labeled test data. We can always compare our model performance on that!
- In real-world scenarios without dataset, it is still necessary to create a small test set with annotations/labels (following any method we discussed last week - from crowd sourcing to active learning!)

Training a Classifier: Feature Extraction

Let us take simple bag of ngrams features for now.



```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(ngram_range=(1, 5))
X_train = vectorizer.fit_transform(df_train_filtered.text.tolist())
X_test = vectorizer.transform(df_test.text.tolist())

print(X_train.shape)
print(X_test.shape)
```

```
(1320, 56734)
(250, 56734)
```

Note: There are several things I am not doing here e.g., removing stop words, or lower casing etc.

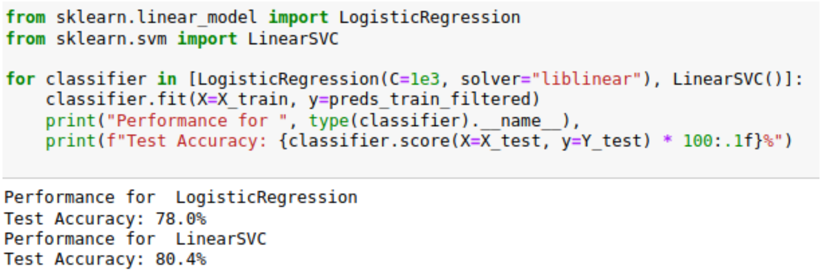
before going ahead with building a model

- Remember that the label model gives out probabilistic labels (i.e., label distribution instead of one label).
- While there are classifiers that work with that format, many classifier algorithms expect one label.
- We can just use the class that got maximum probability from this distribution, using existing function in snorkel.

```
from snorkel.utils import probs_to_preds
preds_train_filtered = probs_to_preds(probs=probs_train_filtered)
```

Now, we can just use any classifier like in the sentiment classification example from NLP Pipeline session.

Classification Performance



```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

for classifier in [LogisticRegression(C=1e3, solver="liblinear"), LinearSVC()]:
    classifier.fit(X=X_train, y=preds_train_filtered)
    print("Performance for ", type(classifier).__name__),
    print(f"Test Accuracy: {classifier.score(X=X_test, y=Y_test) * 100:.1f}%")
```

```
Performance for  LogisticRegression
Test Accuracy: 78.0%
Performance for  LinearSVC
Test Accuracy: 80.4%
```

So, wait ... didn't the LabelModel/MajorityModel do much better than this?? Why are we doing this?

Label Model vs Classifier

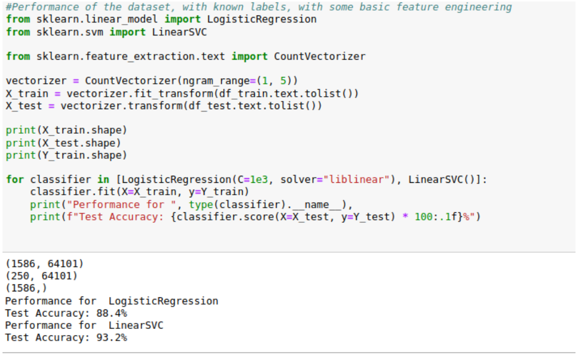
- What is the difference?
- Why can't we just use that label model itself as a classifier- it seems to be doing some sort of classifciation?
- Why am I getting poorer result with the classifier here?
- Why can't I just use LFs as features, instead of Bag of Words/ngrams etc?

Is this good show?

- 80% accuracy doesn't seem that bad to me, since randomly predicting SPAM for everything would have given closer to 50% accuracy on this dataset. (check the dataset stats!)
- However, we know the original labels too in this case. How does this approach compare with that?

Is this good show?

- 80% accuracy doesn't seem that bad to me, since randomly predicting SPAM for everything would have given closer to 50% accuracy on this dataset. (check the dataset stats!)
- However, we know the original labels too in this case. How does this approach compare with that?



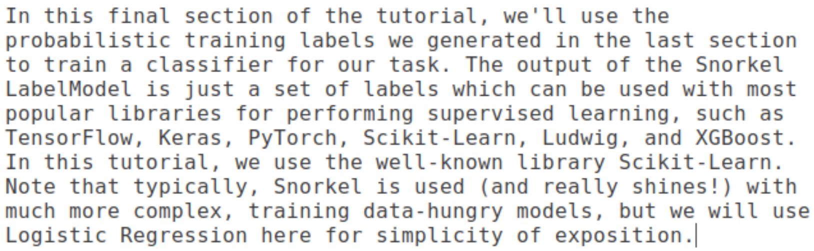
```
#Performance of the dataset, with known labels, with some basic feature engineering
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(ngram_range=(1, 5))
X_train = vectorizer.fit_transform(df_train.text.tolist())
X_test = vectorizer.transform(df_test.text.tolist())

print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)

for classifier in [LogisticRegression(C=1e3, solver="liblinear"), LinearSVC()]:
    classifier.fit(X=X_train, y=Y_train)
    print("Performance for ", type(classifier).__name__),
    print(f"Test Accuracy: {classifier.score(X=X_test, y=Y_test) * 100:.1f}%")
```

```
(1586, 64101)
(250, 64101)
(1586,)
Performance for  LogisticRegression
Test Accuracy: 88.4%
Performance for  LinearSVC
Test Accuracy: 93.2%
```

So why bother about snorkel?

- Remember, in actual use cases of snorkel, we don't know training labels!
- We have to come up with a reasonably good model that does well on test data, WITHOUT labeled training data.
- With simple, not so well thought out LFs, we already got 80% accuracy! May be better LFs will give better results!

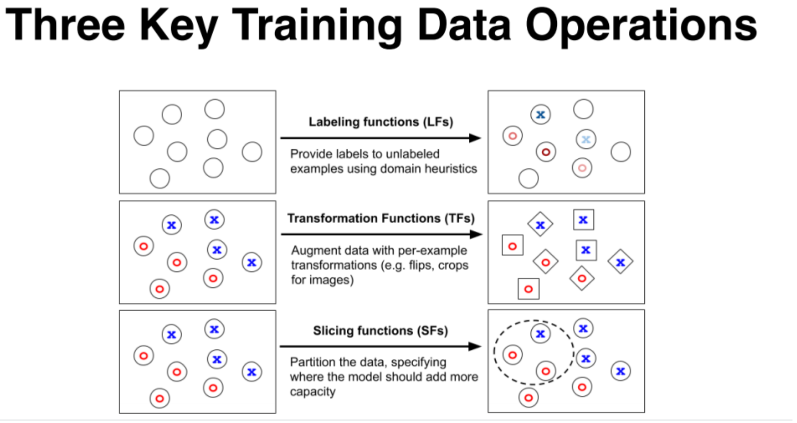
Snorkel disclaimer


In this final section of the tutorial, we'll use the probabilistic training labels we generated in the last section to train a classifier for our task. The output of the Snorkel LabelModel is just a set of labels which can be used with most popular libraries for performing supervised learning, such as TensorFlow, Keras, PyTorch, Scikit-Learn, Ludwig, and XGBoost. In this tutorial, we use the well-known library Scikit-Learn. Note that typically, Snorkel is used (and really shines!) with much more complex, training data-hungry models, but we will use Logistic Regression here for simplicity of exposition.

There is a reason why I chose to show this here, instead of somewhere else during the session.

Transformation Functions

How to "create" data?


Three Key Training Data Operations
Labeling functions (LFs)
Provide labels to unlabeled examples using domain heuristics
Transformation Functions (TFs)
Augment data with per-example transformations (e.g. flips, crops for images)
Slicing functions (SFs)
Partition the data, specifying where the model should add more capacity

Data Augmentation

- Data augmentation is a popular technique for increasing the size of labeled training sets
- In this, we create copies of labeled data points. by transforming the text, but preserving the class/category information.
- This is very common in work on image classification etc, and quickly becoming popular in NLP too.

Data Augmentation in Snorkel

- Accomplished through "Transformation Functions" (@transformation_function decorator in Python)
- Transformation functions are functions that can be applied to a training data point to create another valid training data point of the same class.
- Transformation functions should be atomic e.g. a small rotation of an image, or changing a single word in a sentence.
- We then compose multiple transformation functions when applying them to training data points.
- Common ways to augment text includes replacing words with their synonyms, or replacing names entities with other entities.
- Our basic modeling assumption is that applying these operations to a comment generally shouldn't change whether it is SPAM or not.

Step 1: Loading the Dataset

Note: We work with labeled set here! Our goal is different from earlier! (Why?)

```
df_train, df_test = load_spam_dataset(load_train_labels=True)
Y_train = df_train["label"].values
Y_test = df_test["label"].values
```

TFs Examples -1

```python
from snorkel.preprocess.nlp import SpacyPreprocessor

spacy = SpacyPreprocessor(text_field="text", doc_field="doc", memoize=True)

import names
from snorkel.augmentation import transformation_function

# Pregenerate some random person names to replace existing ones with
# for the transformation strategies below
replacement_names = [names.get_full_name() for _ in range(50)]


# Replace a random named entity with a different entity of the same type.
@transformation_function(pre=[spacy])
def change_person(x):
    person_names = [ent.text for ent in x.doc.ents if ent.label_ == "PERSON"]
    # If there is at least one person name, replace a random one. Else return None.
    if person_names:
        name_to_replace = np.random.choice(person_names)
        replacement_name = np.random.choice(replacement_names)
        x.text = x.text.replace(name_to_replace, replacement_name)
        return x


# Swap two adjectives at random.
@transformation_function(pre=[spacy])
def swap_adjectives(x):
    adjective_idxs = [i for i, token in enumerate(x.doc) if token.pos_ == "ADJ"]
    # Check that there are at least two adjectives to swap.
    if len(adjective_idxs) >= 2:
        idx1, idx2 = sorted(np.random.choice(adjective_idxs, 2, replace=False))
        # Swap tokens in positions idx1 and idx2.
        x.text = " ".join(
            [
                x.doc[:idx1].text,
                x.doc[idx2].text,
                x.doc[1 + idx1 : idx2].text,
                x.doc[idx1].text,
                x.doc[1 + idx2 :].text,
            ]
        )
        return x
```

TFs Examples -2

```python
import nltk
from nltk.corpus import wordnet as wn

nltk.download("wordnet")


def get_synonym(word, pos=None):
    """Get synonym for word given its part-of-speech (pos)."""
    synsets = wn.synsets(word, pos=pos)
    # Return None if wordnet has no synsets (synonym sets) for this word and pos.
    if synsets:
        words = [lemma.name() for lemma in synsets[0].lemmas()]
        if words[0].lower() != word.lower():  # Skip if synonym is same as word.
            # Multi word synonyms in wordnet use '_' as a separator e.g. reckon_with. Replace it with space.
            return words[0].replace("_", " ")


def replace_token(spacy_doc, idx, replacement):
    """Replace token in position idx with replacement."""
    return " ".join([spacy_doc[:idx].text, replacement, spacy_doc[1 + idx :].text])
```

TFs Examples -3

```
@transformation_function(pre=[spacy])
def replace_verb_with_synonym(x):
    # Get indices of verb tokens in sentence.
    verb_idxs = [i for i, token in enumerate(x.doc) if token.pos_ == "VERB"]
    if verb_idxs:
        # Pick random verb idx to replace.
        idx = np.random.choice(verb_idxs)
        synonym = get_synonym(x.doc[idx].text, pos="v")
        # If there's a valid verb synonym, replace it. Otherwise, return None.
        if synonym:
            x.text = replace_token(x.doc, idx, synonym)
            return x


@transformation_function(pre=[spacy])
def replace_noun_with_synonym(x):
    # Get indices of noun tokens in sentence.
    noun_idxs = [i for i, token in enumerate(x.doc) if token.pos_ == "NOUN"]
    if noun_idxs:
        # Pick random noun idx to replace.
        idx = np.random.choice(noun_idxs)
        synonym = get_synonym(x.doc[idx].text, pos="n")
        # If there's a valid noun synonym, replace it. Otherwise, return None.
        if synonym:
            x.text = replace_token(x.doc, idx, synonym)
            return x


@transformation_function(pre=[spacy])
def replace_adjective_with_synonym(x):
    # Get indices of adjective tokens in sentence.
    adjective_idxs = [i for i, token in enumerate(x.doc) if token.pos_ == "ADJ"]
    if adjective_idxs:
        # Pick random adjective idx to replace.
        idx = np.random.choice(adjective_idxs)
        synonym = get_synonym(x.doc[idx].text, pos="a")
        # If there's a valid adjective synonym, replace it. Otherwise, return None.
        if synonym:
            x.text = replace_token(x.doc, idx, synonym)
            return x
```

TFs Result

```
tfs = [
    change_person,
    swap_adjectives,
    replace_verb_with_synonym,
    replace_noun_with_synonym,
    replace_adjective_with_synonym,
]
from utils import preview_tfs
import numpy as np

preview_tfs(df_train, tfs)
```

	TF Name	Original Text	Transformed Text
0	change_person	Check out Berzerk video on my channel ! :D	Check out Doris Masterson video on my channel ! :D
1	swap_adjectives	hey guys look im aware im spamming and it pisses people off but please take a moment to check out my music. im a young rapper and i love to do it and i just wanna share my music with more people just click my picture and then see if you like my stuff	hey guys look im more im spamming and it pisses people off but please take a moment to check out my music. im a young rapper and i love to do it and i just wanna share my music with aware people just click my picture and then see if you like my stuff
2	replace_verb_with_synonym	"eye of the tiger" "i am the champion" seems like katy perry is using titles of old rock songs for lyrics..	"eye of the tiger" "i am the champion" seems like katy perry be using titles of old rock songs for lyrics..
3	replace_noun_with_synonym	"eye of the tiger" "i am the champion" seems like katy perry is using titles of old rock songs for lyrics..	"eye of the tiger" "i am the champion" seems like katy perry is using title of old rock songs for lyrics..
4	replace_adjective_with_synonym	You gotta say its funny. well not 2 billion worth funny but still. It clicked and everything went uphill. At least you don't have JB's shit on #1.	You gotta say its funny. well not 2 billion worth funny but still. It clicked and everything went acclivitous . At least you don't have JB's shit on #1.

Notes on TFs

- We notice that some changes are trivial.
- Sometimes they introduce incorrect grammar to the sentence (e.g. swap_adjectives swapping "young" and "more" above).

Notes on TFs

- We notice that some changes are trivial.
- Sometimes they introduce incorrect grammar to the sentence (e.g. swap_adjectives swapping "young" and "more" above).
- This can still be helpful for training our model, because it teaches the model to be invariant to such small changes.

Notes on TFs

- We notice that some changes are trivial.
- Sometimes they introduce incorrect grammar to the sentence (e.g. swap_adjectives swapping "young" and "more" above).
- This can still be helpful for training our model, because it teaches the model to be invariant to such small changes.
- The TFs are expected to be heuristic strategies that indeed preserve the class most of the time, but don't need to be perfect.
- Snorkel is built to work with such noisy output

Before Applying TFs on our dataset

- There are so many of these. Should we apply everything on all data points? Is that necessary? Is it useful?

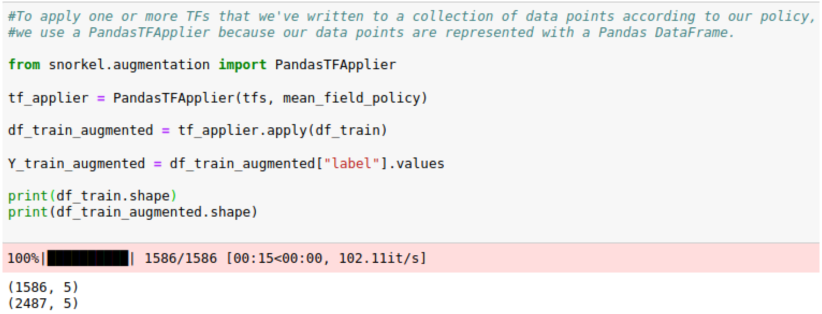
Before Applying TFs on our dataset

- There are so many of these. Should we apply everything on all data points? Is that necessary? Is it useful?
- Snorkel has some data augmentation policies to choose from.
- RandomPolicy: Samples sequences of TF indices a specified length at random from the total number of TFs.
- MeanFieldPolicy: When we know some TFs are important/useful than others.

TFs policies

```
from snorkel.augmentation import RandomPolicy
random_policy = RandomPolicy(len(tfs), sequence_length=2,
                n_per_original=2, keep_original=True)

from snorkel.augmentation import MeanFieldPolicy

mean_field_policy = MeanFieldPolicy(len(tfs),
                    sequence_length=2,
                    n_per_original=2,
                    keep_original=True,
                    p=[0.05, 0.05, 0.3, 0.3, 0.3],)
```

Applying TFs



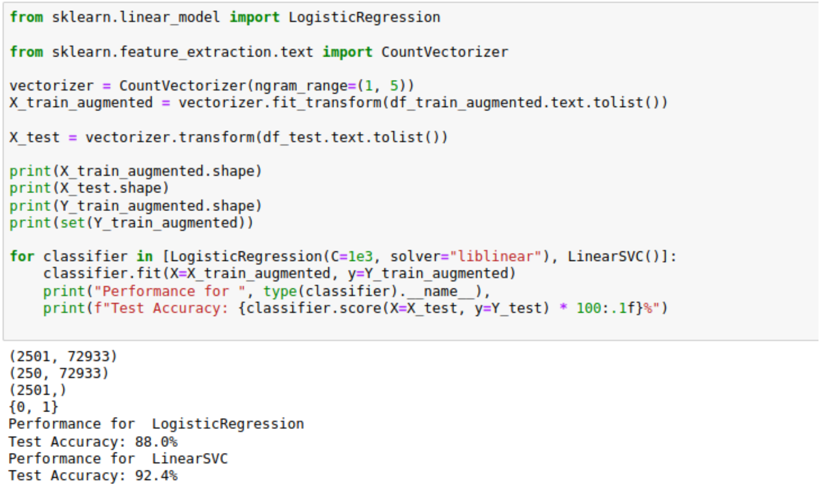
```
#To apply one or more TFs that we've written to a collection of data points according to our policy,
#we use a PandasTFApplier because our data points are represented with a Pandas DataFrame.

from snorkel.augmentation import PandasTFApplier

tf_applier = PandasTFApplier(tfs, mean_field_policy)

df_train_augmented = tf_applier.apply(df_train)

Y_train_augmented = df_train_augmented["label"].values

print(df_train.shape)
print(df_train_augmented.shape)
```

```
100%|██████████| 1586/1586 [00:15<00:00, 102.11it/s]
```

```
(1586, 5)
(2487, 5)
```

So we almost doubled the size of our dataset.

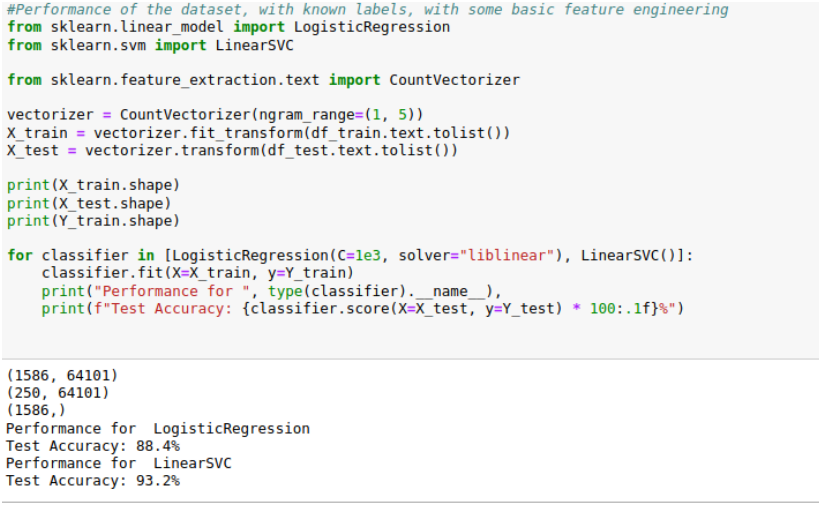
Training the classifier



```
from sklearn.linear_model import LogisticRegression

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(ngram_range=(1, 5))
X_train_augmented = vectorizer.fit_transform(df_train_augmented.text.tolist())

X_test = vectorizer.transform(df_test.text.tolist())

print(X_train_augmented.shape)
print(X_test.shape)
print(Y_train_augmented.shape)
print(set(Y_train_augmented))

for classifier in [LogisticRegression(C=1e3, solver="liblinear"), LinearSVC()]:
    classifier.fit(X=X_train_augmented, y=Y_train_augmented)
    print("Performance for ", type(classifier).__name__),
    print(f"Test Accuracy: {classifier.score(X=X_test, y=Y_test) * 100:.1f}%")
```

```
(2501, 72933)
(250, 72933)
(2501,)
{0, 1}
Performance for  LogisticRegression
Test Accuracy: 88.0%
Performance for  LinearSVC
Test Accuracy: 92.4%
```

How was it before data augmentation?



```
#Performance of the dataset, with known labels, with some basic feature engineering
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(ngram_range=(1, 5))
X_train = vectorizer.fit_transform(df_train.text.tolist())
X_test = vectorizer.transform(df_test.text.tolist())

print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)

for classifier in [LogisticRegression(C=1e3, solver="liblinear"), LinearSVC()]:
    classifier.fit(X=X_train, y=Y_train)
    print("Performance for ", type(classifier).__name__),
    print(f"Test Accuracy: {classifier.score(X=X_test, y=Y_test) * 100:.1f}%")
```

```
(1586, 64101)
(250, 64101)
(1586,)
Performance for  LogisticRegression
Test Accuracy: 88.4%
Performance for  LinearSVC
Test Accuracy: 93.2%
```

Whaat?

- All this work just to get a slight drop in performance compared to original?? (yeah, I know.. that sucks..)
- But the original snorkel tutorial shows a 5% increase due to data augmentation!! (they directly trained a neural model, and it was bad to start with, so augmentation made it better).
- What does that tell you?:

Whaat?

- All this work just to get a slight drop in performance compared to original?? (yeah, I know.. that sucks..)
- But the original snorkel tutorial shows a 5% increase due to data augmentation!! (they directly trained a neural model, and it was bad to start with, so augmentation made it better).
- What does that tell you?:
  - We don't have to jump to neural networks right away - there may be *simpler/lighter* solutions that work equally well or better.
  - Data augmentation is not guaranteed to solve all your model performance problems!

Assignment 2 Description

- 1 question, 15% of the grade.
- Write a rule based NLP system that creates a small corpus of sentences indicating birth/death information of people from Wikipedia biography pages (from the text, not from the info box on the right!) such as place/year of birth/death.
- You can use any language (human/programming).
- You can take a look at information extraction tutorial on snorkel for ideas.
- Write a report describing your approach, how much data you collected etc., and submit along with code.

Next Class

- Different kinds of neural text embeddings in NLP
- How can we use them for NLP
- How can we use them for fine tuning
- To Do for you: group presentation prep, going through class materials and asking questions on forum.

Again, I will upload today's code in a day or two after cleaning it up.