Data Programming Creating Large Training sets, Quickly

Presenting by

Shreeya Badhe
Varsha Chandrahasareddy Mulangi
Akash Yadav Muniraju

Challenges in creating large labeled dataset

- Time consuming
- Expensive as we require domain experts for long duration

Solution

Data Programming

- Programmatic approach for creating large training sets quickly.

Uses Weak Supervision technique for labeling functions

- Heuristics
- Rules

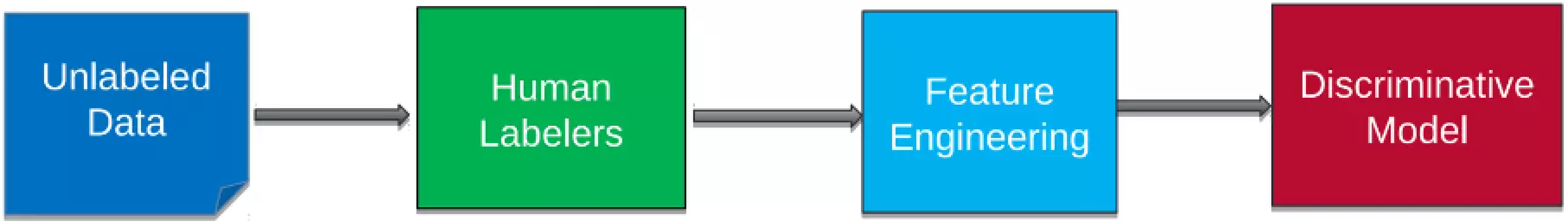
Example of a Heuristic

LF1: Label as "Technology" if the article contains the word "iPhone".

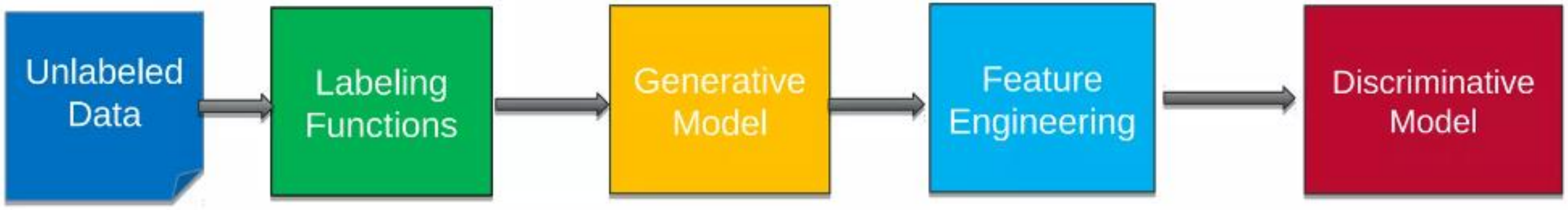
Heuristic: Articles mentioning "iPhone" are likely about technology.

Pipeline

- Traditional Supervision



- Data Programming



Training Set

a) Exploring the training set for initial ideas

We'll start by looking at 20 random data points from the `train` set to generate some ideas for LFs.

[6]:
```
df_train[["author", "text", "video"]].sample(20, random_state=2)
```

[6]:

	author	text	video
4	ambareesh nimkar	"eye of the tiger" "i am the champion" seems l...	2
87	pratik patel	mindblowing dance.,.,.superbbb song	3
14	RaMpAgE420	Check out Berzerk video on my channel ! :D	4
80	Jason Haddad	Hey, check out my new website!! This site is a...	1
104	austin green	Eminem is my insperasen and fav	4
305	M.E.S	hey guys look im aware im spamming and it piss...	4
22	John Monster	Oh my god ... Roar is the most liked video at ...	2
338	Alanoud Alsaleh	I started hating Katy Perry after finding out ...	2
336	Leonardo Baptista	http://www.avaaz.org/po/petition/Youtube_Corpo...	1

Labeling Functions

```python
from snorkel.labeling import labeling_function


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


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

Label Matrix

```
from snorkel.labeling import PandasLFApplier

lfs = [check_out, check]

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

```
100%|████████████████████████████████████████████████████████████████| 1586/1586 [00:00<00:00, 27057.27it/s]
```

```
L_train
```

```
array([[-1, -1],
       [-1, -1],
       [-1,  1],
       ...,
       [ 1,  1],
       [-1,  1],
       [ 1,  1]])
```

Coverage Values of Check and Check_out

```
coverage_check_out, coverage_check = (L_train != ABSTAIN).mean(axis=0)
print(f"check_out coverage: {coverage_check_out * 100:.1f}%")
print(f"check coverage: {coverage_check * 100:.1f}%")
```

```
check_out coverage: 21.4%
check coverage: 25.8%
```

Labeling Functions

```
from snorkel.labeling import LabelingFunction


def keyword_lookup(x, keywords, label):
    if any(word in x.text.lower() for word in keywords):
        return label
    return ABSTAIN


def make_keyword_lf(keywords, label=SPAM):
    return LabelingFunction(
        name=f"keyword_{keywords[0]}",
        f=keyword_lookup,
        resources=dict(keywords=keywords, label=label),
    )


"""Spam comments talk about 'my channel', 'my video', etc."""
keyword_my = make_keyword_lf(keywords=["my"])

"""Spam comments ask users to subscribe to their channels."""
keyword_subscribe = make_keyword_lf(keywords=["subscribe"])

"""Spam comments post links to other channels."""
keyword_link = make_keyword_lf(keywords=["http"])
```

Heuristics

```python
@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
```

Combining Labeling Function Outputs with the Label Model

```python
lfs = [
    keyword_my,
    keyword_subscribe,
    keyword_link,
    keyword_please,
    keyword_song,
    regex_check_out,
    short_comment,
    has_person_nlp,
    textblob_polarity,
    textblob_subjectivity,
]
```

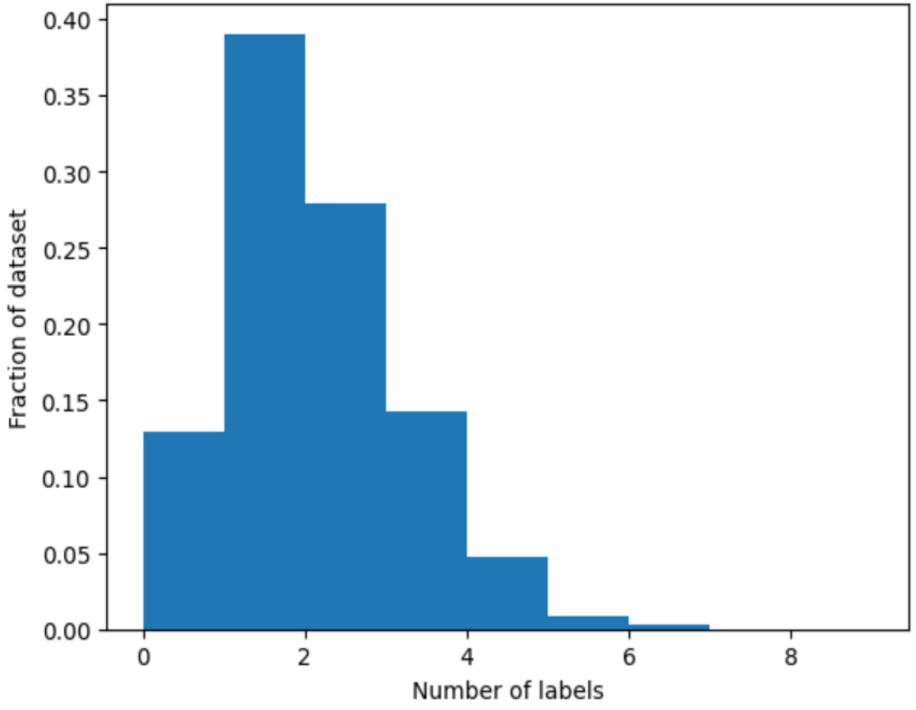
Output

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

	j	Polarity	Coverage	Overlaps	Conflicts
keyword_my	0	[1]	0.198613	0.185372	0.109710
keyword_subscribe	1	[1]	0.127364	0.108449	0.068726
keyword_http	2	[1]	0.119168	0.100252	0.080706
keyword_please	3	[1]	0.112232	0.109710	0.056747
keyword_song	4	[0]	0.141866	0.109710	0.043506
regex_check_out	5	[1]	0.233922	0.133039	0.087011
short_comment	6	[0]	0.225725	0.145019	0.074401
has_person_nlp	7	[0]	0.071879	0.056747	0.030895
textblob_polarity	8	[0]	0.035309	0.032156	0.005044
textblob_subjectivity	9	[0]	0.357503	0.252837	0.160151

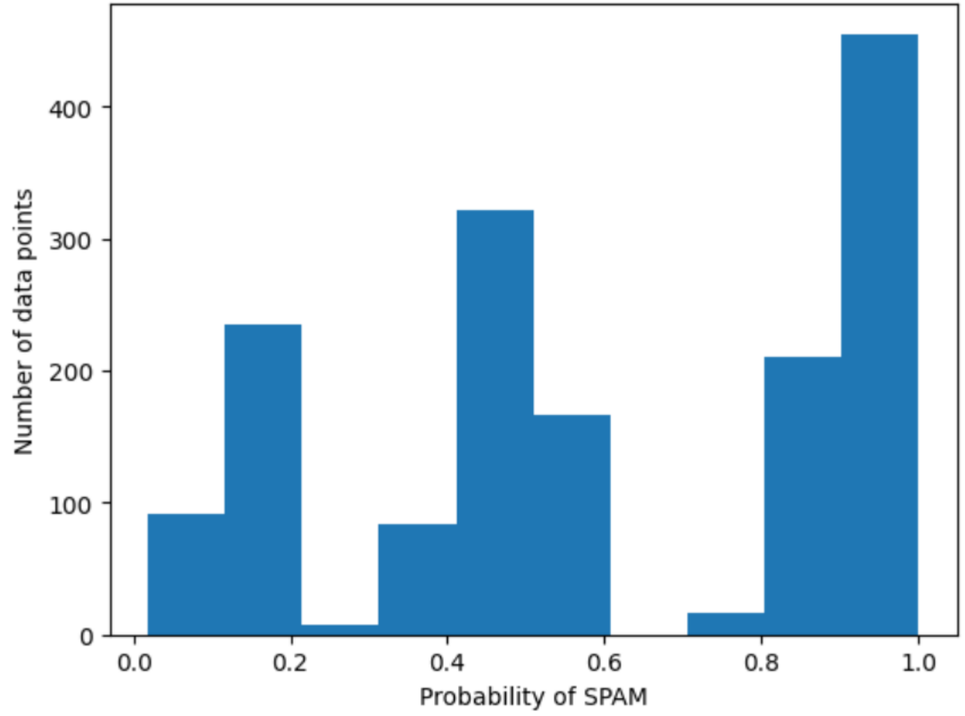
Histogram showing how many LF labels the data points in our train set have to get an idea of our total coverage.

```
import matplotlib.pyplot as plt

%matplotlib inline


def plot_label_frequency(L):
    plt.hist((L != ABSTAIN).sum(axis=1), density=True, bins=range(L.shape[1]))
    plt.xlabel("Number of labels")
    plt.ylabel("Fraction of dataset")
    plt.show()


plot_label_frequency(L_train)
```



Histogram shows the confidences we have that each data point has the label SPAM.

```python
def plot_probabilities_histogram(Y):
    plt.hist(Y, bins=10)
    plt.xlabel("Probability of SPAM")
    plt.ylabel("Number of data points")
    plt.show()


probs_train = label_model.predict_proba(L=L_train)
plot_probabilities_histogram(probs_train[:, SPAM])
```



The points we are least certain about will have labels close to 0.5.

Training a Classifier

Featurization

```python
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())
```

Scikit-Learn Classifier

```python
from snorkel.utils import probs_to_preds

preds_train_filtered = probs_to_preds(probs=probs_train_filtered)
```

We then use these labels to train a classifier as usual.

```python
from sklearn.linear_model import LogisticRegression

sklearn_model = LogisticRegression(C=1e3, solver="liblinear")
sklearn_model.fit(X=X_train, y=preds_train_filtered)
```

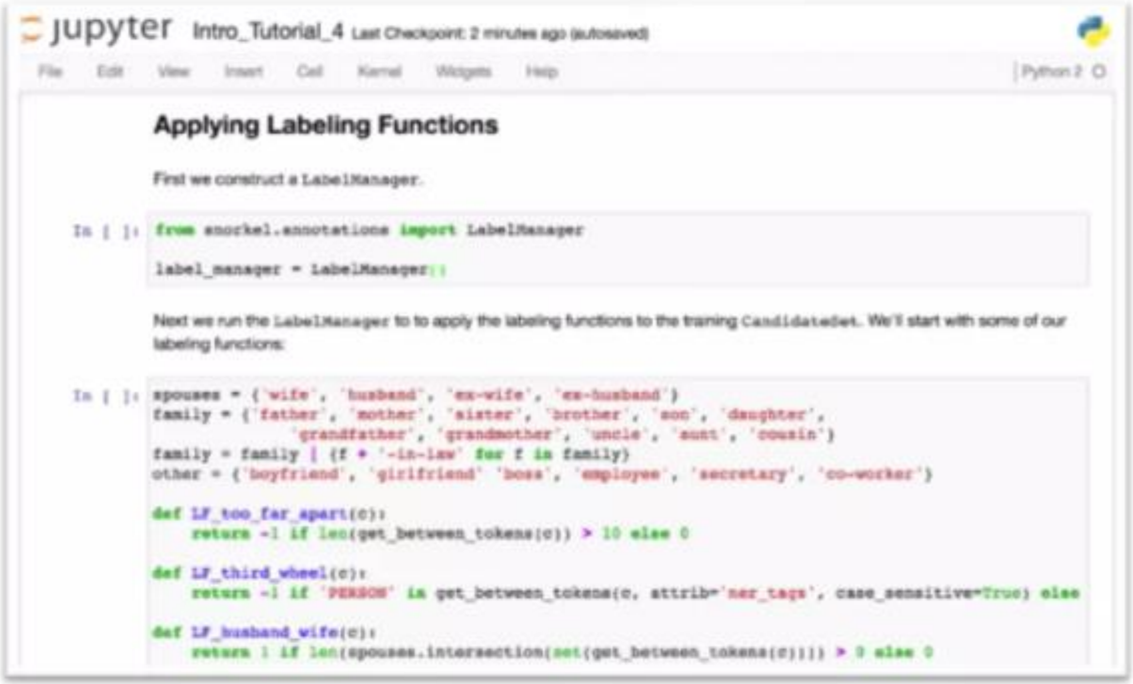
```
LogisticRegression(C=1000.0, solver='liblinear')
```

```python
print(f"Test Accuracy: {sklearn_model.score(X=X_test, y=Y_test) * 100:.1f}%")
```

```
Test Accuracy: 94.4%
```

Snorkel

- Snorkel is a library developed at Stanford for programmatically building and managing training datasets.



Thank you ☺