shed

A FRAMEWORK FOR REPRODUCABLE TEXT MINING RESEARCH

15 October, 2015

Chris Emmery, Walter Daelemans

Universiteit Antwerpen

CLiPS

SETTING: CODECAMP

- 3 days of author profiling with CLiPS a.o.*.
- Standard tasks: prepare data, discuss features, try classifiers.
- Missing: general framework to uniformely report and compare pipeline performance across tasks and datatset combinations.

*Walter Daelemans, Guy De Pauw, Mike Kestemont, Tom De Smedt, Ben Verhoeven, Janneke van de Loo, Chris Emmery, Enrique Manjavacas, Florian Kunneman.

GENERAL IDEA

- Usable as a working environment as well as an (almost) end-to-end system.
- Flexible enough for extention and adaptation to different tasks.
- Open backbone, implementation for Frog (LaMachine) and spaCy.
- Report settings across the entire pipeline.
- Save backbone settings, fitted featurizers, and trained classifiers for easy deployment and reproduction.
- Well documented and open-source.

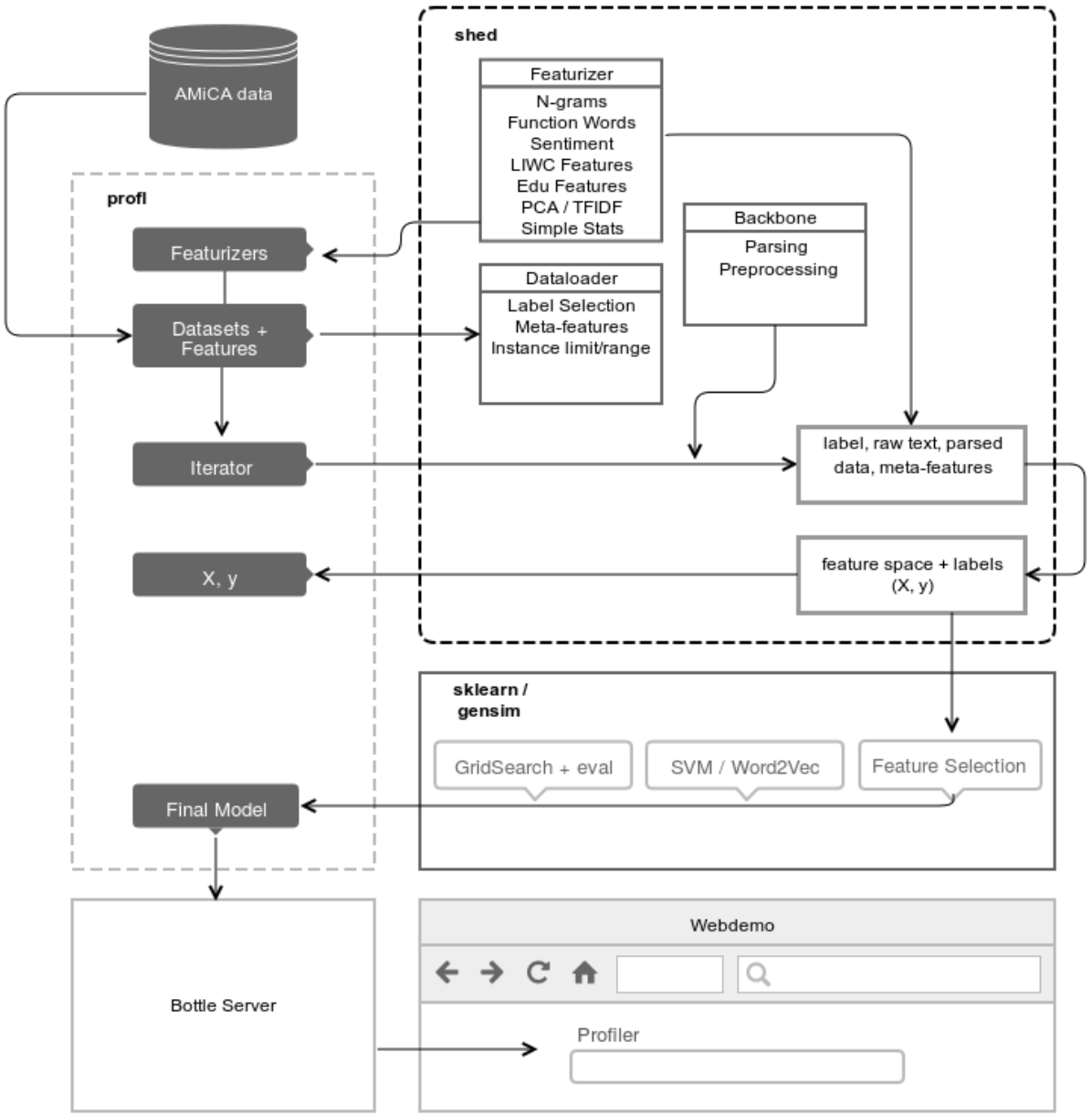
ARCHITECTURE

- `experiment.py` - wrapping and reporting standard procedures.
- `environment.py` - saving and loading progress.
- `datareader.py` - iterative loading of data in standard format.
- `backbone.py` - preprocessing according to own preference.
- `featurizer.py` - skeleton and implementations for feature extraction.

FEATURIZERS - STANDARD IMPLEMENTATIONS

- `SimpleStats`
  - flooding, character types, number emoticons, mentions, hashtags
  - average word length, caps, start caps, urls, photos, videos
- `Ngrams`
  - n amount of: characters, tokens, POS (up to max features)
  - relative & cutoff on frequency
- `Function Words`
- `SentimentFeatures`
- `Readability`

SCHEMA


shed
AMiCA data
Featurizer
N-grams
Function Words
Sentiment
LIWC Features
Edu Features
PCA / TFIDF
Simple Stats
Backbone
Parsing
Preprocessing
Dataloader
Label Selection
Meta-features
Instance limit/range
profl
Featurizers
Datasets + Features
Iterator
X, y
label, raw text, parsed data, meta-features
feature space + labels (X, y)
sklearn / gensim
GridSearch + eval
SVM / Word2Vec
Feature Selection
Final Model
Bottle Server
Webdemo
Profiler

ENVIRONMENT

```
1  import shed
2  from shed.featurizer import Ngrams, SentimentFeatures, FuncWords
3  from sklearn.svm import SVC
4  from sklearn import metrics
5
6  data = ['/home/chris/data/netlog.csv', '/home/chris/data/otherstuff.csv']
7  features = [
8      Ngrams(level='token', n_list=[1]),
9      Ngrams(level='char', n_list=[2, 3]),
10     SentimentFeatures(),
11     FuncWords()
12 ]
13
14 env = shed.Environment(name, backbone='frog')
15
16 ftf_loader = env.load(data=data[0], target_label='age', proc='both', max_n=5000)
17 X, y = env.fit_transform(ftf_loader, features)
18
19 trf_loader = env.load(data=data[1], target_label='age', proc='both', max_n=1000)
20 Xi, yi = env.transform(trf_loader)
21
22 env.train(SVC(), X, y)
23 res = env.classify(clf, Xi, yi)
24
25 print(metrics.f1_score(yi, res, pos_label=1, average='binary'))
26 env.save()
```

EXPERIMENT.PY

```
...

if __name__ == '__main__':

    features = [
        Ngrams(level='token', n_list=[1]),
        Ngrams(level='char', n_list=[2, 3])
    ]

    def dr(a): return '/home/chris/data/{0}.csv'.format(a)
    exp = {
                        # data             instances    config  balance  params
            'CV(X)':     ([dr('netlog')], (1000, 5000), False,  False, ('rbf', 1e-3, 500)),
            'X->Xi':     ([dr('netlog'),
                           dr('stuff')],  (-1, -1),     'trte', True,  None),
            'CVGrid(X)': ([dr('netlog')], (1000, 1000), 'grid', False, False)
        }

    for k, v in exp.items():
        clf = svm.SVC
        name, features, instances, config, balance, params = (k,) + v
        experiment(name, features, instances, config, balance, clf, params)
```

LOG

```
---- Shed ----

 Config:

        feature:   token_ngram
        n_list:    [1, 2]
        max_feat:  None


        feature:   char_ngram
        n_list:    [3]
        max_feat:  None

 name: somename
 seed: 123
 clasf: gender
 pos, neg: 1000, 1000

  Reading from /home/chris/data/netlog.csv...
  Acquired (1000, 1000) from data.
  Reading from /home/chris/data/stuff.csv...
  Acquired (1000, 1000) from data.
```

LOG

```
Classes: [0 1]

Sparse shape: (2000, 8739)

Total Data instances: 2000
Total Label instances: 2000

Distribution: [(0, 1000), (1, 1000)]
Accuracy @ majority baseline:   0.5

Gridsearch:
{'svc__gamma': 0.01, 'svc__C': 100, 'svc__kernel': 'rbf'}

Tf-CV Result:
0.78565912214803
```