Experiencing NLTK
Classifiers

Stanislav Reichert

Tübingen University

stanislav.reichert@student.uni-tuebingen.de

December 5, 2017

Supervised Machine Learning using NLTK

Classification task:

- Goal: predict the correct label for a given input
- Means: Observable patterns (e.g. correlation between word frequency and text topic)
- Basic strategy: Classifiers rely on training corpora which assign (correct) label to each input from a predefined set
- Challenge: identify relevant features of the input. Then encode them appropriately

On the Way to a Feature Set: Bag of Words

- Bag of words turns a list of words into a dictionary of the form: word:boolean
- Bag of words may ignore the number of occurrences (as in the follwoing examples), the order of words in a sentence and the fact, that some of these words are noise (stopwords, etc.)

```
# A naive Bayesian classifier classifies naive features
{'A': True, 'naive': True, 'features': True,
'classifier': True,  'classifies': True, 'Bayesian': True}

from nltk.corpus import stopwords
...
# Use the respective stopfile, e.g. stopfile='english'

def cleanMyText(wordList, stopfile='english'):
    words_to_delete = stopwords.words(stopfile)
    ...
```

Supervised Machine Learning using NLTK (Cont.)

NLTK defines following classifier classes:

- ConditionalExponentialClassifier
- **NaiveBayesClassifier**
- **DecisionTreeClassifier**
- MaxentClassifier
- WekaClassifier

NLTK provides:

- implementation of these classifiers
- wrappers for Scikit-Learn classifiers

NaiveBayes: Minimal Working Example

```
1   import nltk
2
3   train_set = [
4   (dict(luck="ok", grades="good", charisma="little"), 'pass'),
5   (dict(luck="ok", grades="superb", charisma="absent"), 'pass'),
6   (dict(luck="bad",  grades="abysmal", charisma="strong"), 'fail'),
7   (dict(luck="wow",  grades="good", charisma="strong"), 'pass')]
8
9   test_set = [
10  (dict(luck="ok", grades="superb", charisma="strong")),
11  (dict(luck="bad", grades="good", charisma="absent")),
12  (dict(luck="superb",  grades="abysmal", charisma="little"))]
13
14  classifier = nltk.classify.NaiveBayesClassifier.train(train_set)
15
16  # nltk.classify.accuracy(classifier, test_set)
17  # Cannot be used here
18  print(classifier.classify_many(test_set))
19  classifier.show_most_informative_features()
```

NaiveBayes: Advanced Example (Code)

```
1  import nltk
2  import random
3  from nltk.corpus import movie_reviews
4
5  documents = [(list(movie_reviews.words(fileid)), category)
6               for category in movie_reviews.categories()
7               for fileid in movie_reviews.fileids(category)]
8
9  random.shuffle(documents)
10
11 print(documents[1])
12
13 all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
14 print(all_words.most_common(15))
```

Adapted from: Bird, Klein & Loper (n.d.)

NaiveBayes: Advanced Example (Output)

```
print(documents[1])

(['note', ':', 'some', 'may', 'consider', 'portions', 'of', 'the', 'following',
'text', 'to', 'be', 'spoilers', '.', 'be', 'forewarned', '.', ..., 'the',
'little', 'mermaid', 'is', 'the', 'best', 'film', 'to', 'come', 'out', 'of',
'the', 'disney', "'", 's', 'modern', 'animation', 'renaissance', ',', 'and',
'one', 'of', 'the', 'greatest', 'animated', 'films', 'ever', 'made', '.'], 'pos')

print(all_words.most_common(15))

[(',', 77717), ('the', 76529), ('.', 65876), ('a', 38106), ('and', 35576),
('of', 34123), ('to', 31937), ("'", 30585), ('is', 25195), ('in', 21822),
('s', 18513), ('"', 17612), ('it', 16107), ('that', 15924), ('-', 15595)]
```

NaiveBayes: Advanced Example (Code)

```
1     word_features = list(all_words)[:2000]
2
3     # document_features(document) checks whether each word in the document
4     # is present in the word_features list.
5
6     featuresets = [(document_features(doc), label) for (doc,label) in documents]
7     # where doc is the document file and label is either 'pos' or 'neg'
8
9     training_data = featuresets[200:]
10    test_data = featuresets[:200]
11
12    classifier = nltk.NaiveBayesClassifier.train(training_data)
13    print(nltk.classify.accuracy(classifier, test_data)*100)
14    classifier.show_most_informative_features(7)
```

NaiveBayes: Advanced Example (Output)

```
60 # accuracy of the classifier

Most Informative Features
contains(hatred) = True                pos : neg =  10.6 : 1.0
contains(captures) = True              pos : neg =   7.8 : 1.0
contains(unimaginative) = True         neg : pos =   7.5 : 1.0
contains(laurie) = True                pos : neg =   7.2 : 1.0
contains(stark) = True                 pos : neg =   7.2 : 1.0
contains(mena) = True                  neg : pos =   6.8 : 1.0
contains(dismissed) = True             pos : neg =   6.5 : 1.0
```

DecisionTreeClassifier (Code)

```
1   import nltk
2   from nltk.corpus import brown
3
4   suffix_fdist = nltk.FreqDist()
5
6   for word in brown.words():
7   word = word.lower()
8       suffix_fdist[word[-1:]] += 1
9       suffix_fdist[word[-2:]] += 1
10      suffix_fdist[word[-3:]] += 1
11
12  common_suffixes =
13  [suffix for (suffix, count) in suffix_fdist.most_common(100)]
14
15  def pos_features(word):
16  features = {}
17  for suffix in common_suffixes:
18      features['endswith({})'.format(suffix)] = word.lower().endswith(suffix)
19      return features
```

DecisionTreeClassifier (Code)

```
1     tagged_words = brown.tagged_words(categories='learned')
2     featuresets = [(pos_features(n), g) for (n,g) in tagged_words]
3
4     size = int(len(featuresets) * 0.1)
5     train_set, test_set = featuresets[size:], featuresets[:size]
6
7     classifier = nltk.DecisionTreeClassifier.train(train_set, binary=True,
8     entropy_cutoff=0.8, depth_cutoff=5, support_cutoff=30)
9
10    print(nltk.classify.accuracy(classifier, test_set))
11    print(classifier) # see next slide for the output
12    print(classifier.classify(pos_features('grammar')))
```

Adapted from: Bird, Klein & Loper (n.d.)

DecisionTreeClassifier (Classifier Pseudocode)

```
if endswith(the) == False:
  if endswith(s) == False:
    if endswith(,) == False:
      if endswith(of) == False:
        if endswith(.) == False: return 'NN'
        if endswith(.) != False: return '.'
      if endswith(of) != False: return 'IN'
    if endswith(,) != False: return ','
  if endswith(s) != False:
    if endswith(is) == False:
      if endswith(was) == False:
        if endswith(as) == False: return 'NNS'
        if endswith(as) != False: return 'CS'
      if endswith(was) != False: return 'BEDZ'
    if endswith(is) != False:
      if endswith(his) == False: return 'BEZ'
      if endswith(his) != False: return 'DT'
if endswith(the) != False: return 'AT'
```

Wrap your troubles in NLTK and dream them away

```
1   from nltk.classify.scikitlearn import SklearnClassifier
2   from sklearn.naive_bayes import MultinomialNB, BernoulliNB
3
4   # the training and test data sets are the same as above
5
6   MultinomialNB_classifier = SklearnClassifier(MultinomialNB())
7   MultinomialNB_classifier.train(training_data)
8   print("MultinomialNB accuracy:",
9   nltk.classify.accuracy(MultinomialNB_classifier, testing_data))
10
11  BernoulliNB_classifier = SklearnClassifier(BernoulliNB())
12  BernoulliNB_classifier.train(training_data)
13  print("BernoulliNB accuracy:",
14  nltk.classify.accuracy(BernoulliNB_classifier, testing_data))
```

Saving and Loading Classifiers

```
1   import pickle
2   ...
3   # train the classifier
4   ...
5
6   # To save the trained classifier:
7   classifier_to_save = open("naivebayes.pickle","wb")
8   pickle.dump(classifier, classifier_to_save)
9   classifier_to_save.close()
10
11  #To open the saved classifier:
12  classifier_to_load = open("naivebayes.pickle", "rb")
13  classifier = pickle.load(classifier_to_load)
14  classifier_to_load.close()
```

Metrics

```
1     from nltk import precision
2     from nltk import recall
3
4     ...
5
6     refsets = collections.defaultdict(set)
7     testsets = collections.defaultdict(set)
8
9     for i, (feats, label) in enumerate(featuresets):
10        refsets[label].add(i)
11        observed = classifier.classify(feats)
12        testsets[observed].add(i)
13
14    print('pos precision:', precision(refsets['pos'], testsets['pos']))
15    print('pos recall:', recall(refsets['pos'], testsets['pos']))
16    print('neg precision:', precision(refsets['neg'], testsets['neg']))
17    print('neg recall:', recall(refsets['neg'], testsets['neg']))
```

Bibliography & References

- Hardeniya, N. (2015). NLTK Essentials. Packt Publishing
- Hardeniya, N., & Perkins, J., & Chopra, D., & Joshi, N., & Mathur, I. (2016). Natural Language Processing: Python and NLTK. Packt Publishing
- Perkins, J. (2014). Python 3 Text Processing with NLTK 3 Cookbook (2nd ed). Packt Publishing
- Bird, S. & Klein, E. & Loper, E. (n.d.). Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit. Available from: http://www.nltk.org/book/