

# Python/NLTKでつくる 英語前置詞誤り訂正器

---

奈良先端科学技術大学院大学  
自然言語処理学研究室

スプリングセミナー2012



# What is NLTK?

---

イントロダクション

Natural Language  
Processing with  
Python



# This is NLTK!!!

---

- Natural Language Tool Kit
- Pythonのための自然言語処理ライブラリ
- “Natural Language Processing with Python”  
「入門 自然言語処理」  
NLPの認知度を向上, 門戸を広げた本
- 現在, Python 2.5~2.7 で使うことができる
- 気になったら... <http://www.nltk.org/>

**pip install nltk**

# はじめに：環境構築

---

```
$git clone git://github.com/tuxedocat/ss2012.git  
$cd ss2012
```

- preprocessor.py, prepchecker.py, feature\_extractor.py  
ss2012\_slide, wdiff\_prep, README.md が入ります
- お手元のMacbookには...
  - nltk 2.0.1, Python 2.7.2, iPython 0.12が入っています
  - エディタ等はお好きなものをお使いください

# 大まかな流れ

---

- 前処理：CLCコーパスを処理しやすいように料理する
- 学習データ・テストデータを作成する
- 素性関数をつくる
- 学習事例をつくる
- MaxentClassifier を学習させる
- テストデータに対して評価を行う
- 考察・エラー分析・改良

# Cambridge Learner Corpus (CLC)

---

- Cambridge ESOLが実施している英語試験の回答を集めたもの
  - <http://www.cambridgeesol.org/japan/>
  - 135,000人の学習者
  - 世界190カ国、130の異なる母語話者
  - 一部のみ自由なアクセスが可能（他はCambridge関係者のみ）



UNIVERSITY OF  
CAMBRIDGE



# CLCのファイルを見てみよう

- 人手で作成されたxmlファイル
- 80種類のエラータグ(例: NS type="RT" → 前置詞置換のエラー)

```
1 <?xml version="1.0" encoding="UTF-8"?>$
2 <learner><head sortkey="TR733*0100*2000*01">$
3   <candidate><personnel><language>Spanish</language><age>21-25</age></personnel><score>33.0</score></candidate>$
4   <text>$
5     <answer1>$
6       <question_number>1</question_number>$
7       <exam_score>4.2</exam_score>$
8       <coded_answer>$
9         <p>Dear Helen Ryan,</p>$
10        <p>I have seen your letter <NS type="RT"><i>for</i><c>about</c></NS> Camp California in the U.S.A and I would
11 like to travel in July because I am pretty busy with my studies and I am working with my dad in his office now.</p>$
12        <p>First of all, you state in the letter that the accommodation at Camp California is in tents or log cabins
13 . I would prefer to stay in <NS type="MD"><c>a</c></NS> <NS type="AGN"><i>tents</i><c>tent</c></NS> because I like bei
14 ng in contact with <NS type="UD"><i>the</i></NS> nature <NS type="UN"><i>life</i></NS>.</p>$
15        <p>Secondly, from the list of activities that you <NS type="R"><i>are also state</i><c>provide</c></NS> in t
16 he letter I would prefer swimming because when I was a child I <NS type="RV"><i>practised</i><c>did</c></NS> this spor
17 t for 8 years. In addition, I would prefer photography too because I love taking photos.</p>$
18        <p>Finally, <NS type="RA"><i>it</i><c>I</c></NS> would be grateful if you could clarify what kind of clothes
19 I must <NS type="TV"><i>be taken</i><c>take</c></NS> and how much I will spend during the stay at the Camp.</p>$
20        <p>I look forward to hearing from you</p>$
21        <p>Yours <NS type="DY"><i>sincerelly</i><c>sincerely</c></NS></p>$
22      </coded_answer>$
23    </answer1>$
24    <answer2>$
```

# 前処理

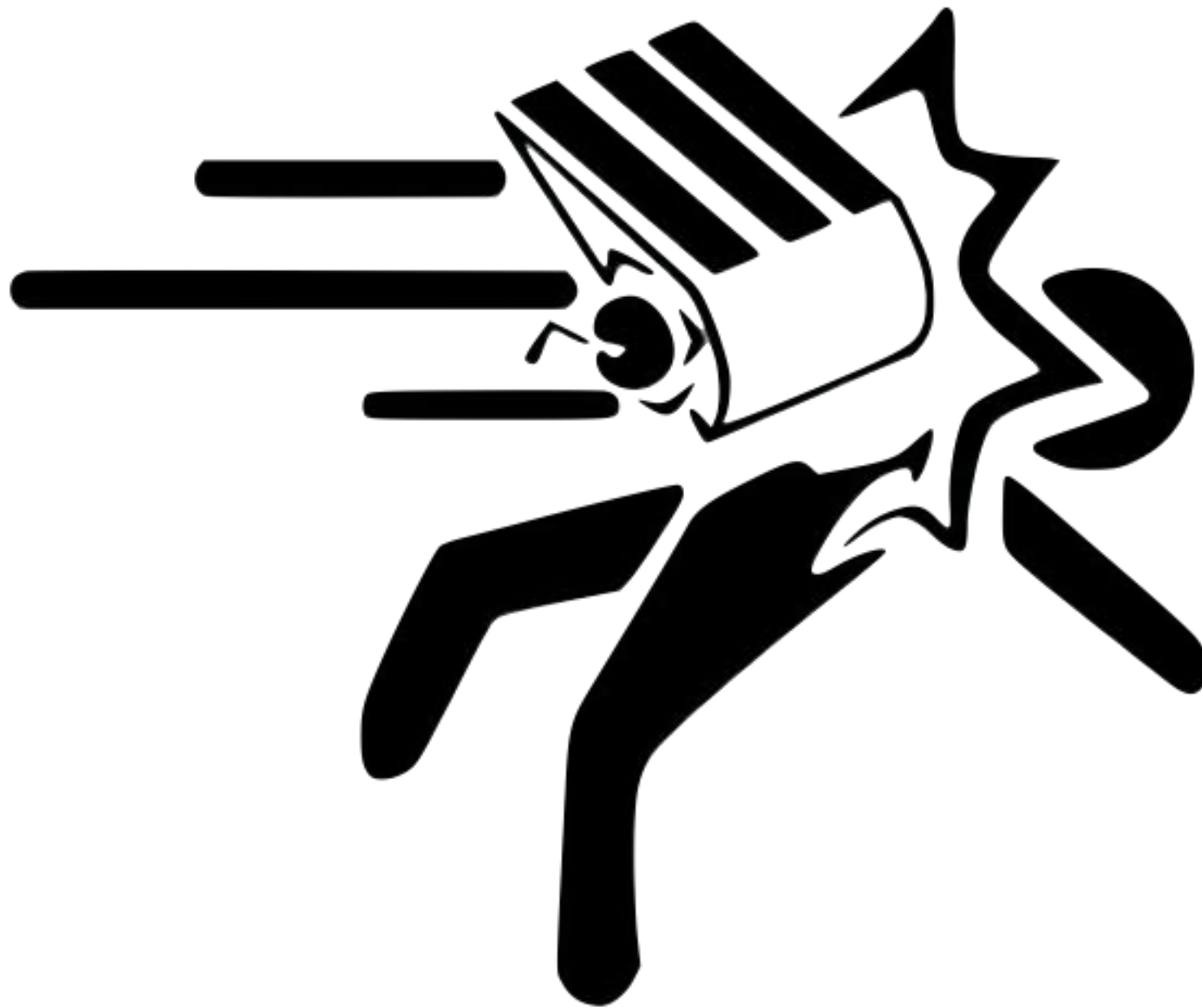
---

- CLCコーパス（を下さしえしたもの： ./wdiff\_prep ファイル）を読み込み，誤り・訂正情報を抽出する

```
clc = open("wdiff_prep", "r").read()
```

- これでリストに入る...
- このままでは段落単位なので，文単位に分割する...
  - NLTKには簡易文分割関数がある(nltk.sent\_tokenize)
- 文(文字列型)から正規表現を用いて訂正情報などを抽出する
- 後で処理しやすいように整形する...





これが意外とめんどくさい

／(^o^)＼ナンテコッタイ

# ...前処理はこちらでやります

---

`$python ./preprocessor.py`

- "packedcorpus.pkl" が出力されます.
  - 今回はコーパスが小さく処理も簡潔なため Pythonのpickleモジュールを使いました.
- ところで...
  - これは実はほんとうの意味での下拵えではないのです...
  - コーパスの整備や前処理はとても大事な工程です!!!!!!
  - ここで現役NLPerの諸氏から色々なエピソードが語られます・・・

# コーパス読み込み

---

- シリアライズされたPython objectを読み込むには...

- Pickle moduleを使う

```
import cPickle as pickle
corpus = pickle.load("packedcorpus.pkl", "rb")
```

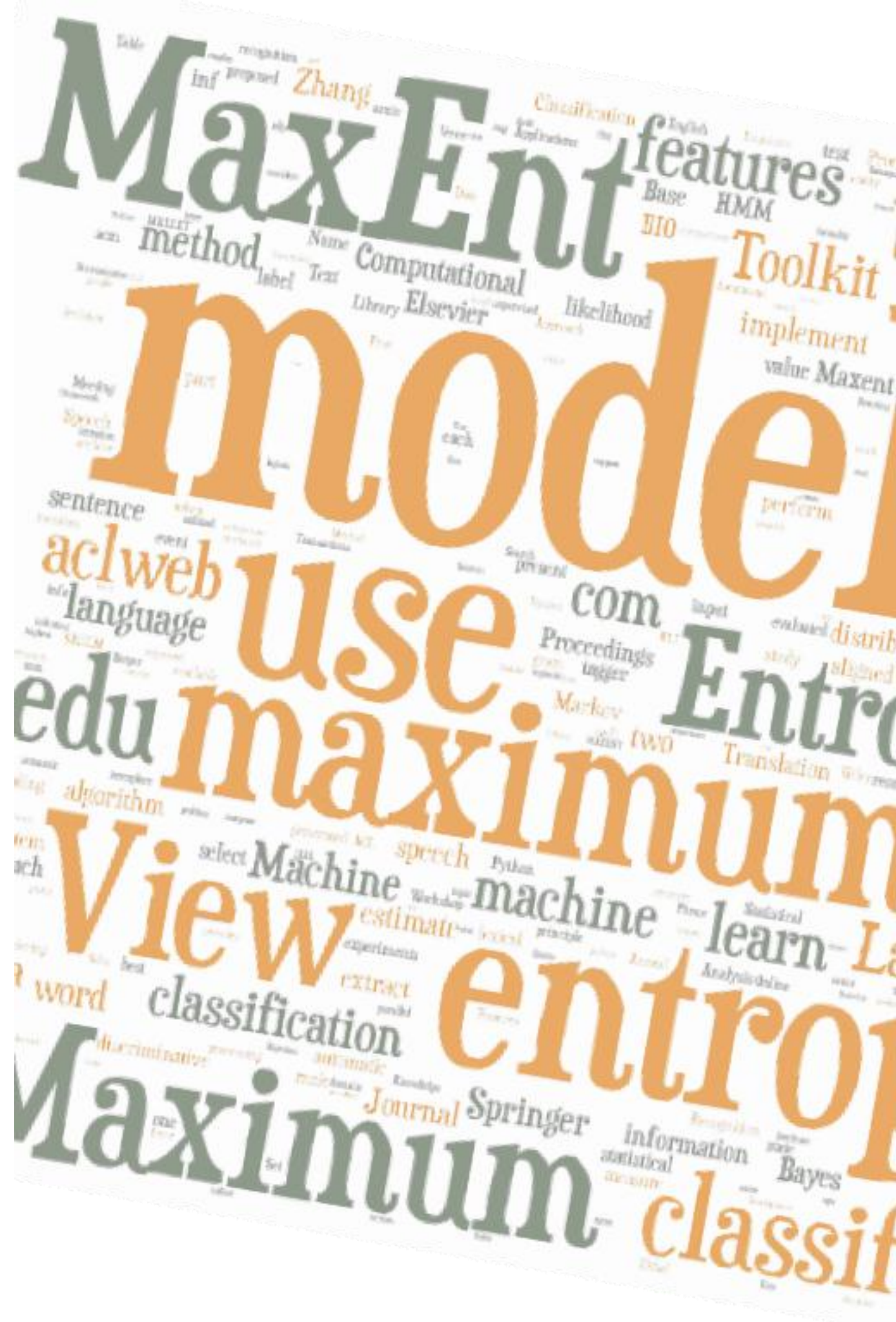
- 一文に対応するdictが入っているlist
  - key "training\_words", "test\_words"  
value = ["tokenized", "sentence", ...]
  - key "correction\_pair", value = ("incorrect prep.", "correct prep.")
  - key "ppindex",  
value = <index of target preposition in tokenized sentence>

# ここからの流れ

---

- コーパスを読み込む
- 素性関数をつくる ← これから
- 学習事例をつくる
- Classifierを学習させる
- テストをする
- 結果を眺める→エラー分析

# nlTK.MaxentClassifierによる 最大エントロピー法分類器 を用いた教師あり分類問題 ...としての前置詞訂正



# 最大エントロピー法：概略

---

- エントロピー：不確実さ、乱雑さの度合い。

- $$H(p) = \sum -P(x) \log P(x)$$

- 不確実な状態ほど、エントロピーは<sup>x</sup>高くなる

- 最大エントロピーの原則：与えられた制約の中で、エントロピーを最大化するモデルを選ぶ。

- 例1：Mさんのじゃんけんパターンをモデル化したい。

- 制約

(1)  $P(\text{グー}) + P(\text{チョキ}) + P(\text{パー}) = 1$ , (2)  $P(\text{グー}) = 0.5$

- 最大エントロピーの原則に従うとすると、どちらのモデルが妥当か。

(ア)  $P(\text{パー}) = 0.1$ ,  $P(\text{チョキ}) = 0.4$ ,  $P(\text{グー}) = 0.5$

(イ)  $P(\text{パー}) = 0.25$ ,  $P(\text{チョキ}) = 0.25$ ,  $P(\text{グー}) = 0.5$

# 最大エントロピー法：例題

---

- 例2: Nさん、Lさん、Pさんが、次のように文を書いたとする。
  - Nさん:  $d_1(\text{block, scissors, block})$ ,  $d_2(\text{paper, scissors, paper})$
  - Lさん:  $d_3(\text{paper, scissors, paper})$ ,  $d_4(\text{block, paper, scissors})$
  - Pさん:  $d_5(\text{block, block, paper})$ ,  $d_6(\text{paper, scissors, block})$
- 今、 $d_x(\text{block, scissors, paper})$ という文が来たとき、Nさん、Lさん、Pさんによって書かれた確率はそれぞれいくらだろうか。



# 最大エントロピー法：例題

---

- 素性(特徴)ベクトルとそのラベル(クラス)を考えてみる

- 例:  $\phi(\phi_1, \phi_2, \dots, \phi_9)$  を次のように定義する

$\phi_1$  = 単語 “block” が1単語目に出現  
 $\phi_2$  = 単語 “block” が2単語目に出現  
 $\phi_3$  = 単語 “block” が3単語目に出現  
 $\phi_4$  = 単語 “paper” が1単語目に出現  
 $\phi_5$  = 単語 “paper” が2単語目に出現  
 $\phi_6$  = 単語 “paper” が3単語目に出現  
 $\phi_7$  = 単語 “scissors” が1単語目に出現  
 $\phi_8$  = 単語 “scissors” が2単語目に出現  
 $\phi_9$  = 単語 “scissors” が3単語目に出現

$d_1 = (1, 0, 1, 0, 0, 0, 0, 1, 0)$ , label: Nさん  
 $d_2 = (0, 0, 0, 1, 0, 1, 0, 1, 0)$ , label: Nさん  
 $d_3 \sim d_6$  を素性ベクトルとラベルで現してみよう。

# 最大エントロピー法：分類・まとめ

---

- 素性とラベルのペアをもとに、新しい文のラベルを推定する。

$$\Pr^* = \arg \max_{\Pr} H(\Pr) = - \arg \max_{\Pr} \sum_{\mathbf{x}, y} \hat{\Pr}[y] \Pr[y|\mathbf{x}] \log \Pr[y|\mathbf{x}]$$

- ※計算は省略しますが気になる人は以下を参考にしてください。
  - 言語処理のための機械学習入門 (高村大也 著)
  - 確率的言語モデル (北研二 著)
  - 朱鷺の杜 (<http://ibisforest.org/index.php?最大エントロピー>)
- 先ほどの例だと、

$$\Pr[N|d_x] > 0.99, \Pr[N|d_x] \ll 0.005, \Pr[N|d_x] \ll 0.005$$

となり、Nさんによって書かれた可能性が高いことがわかる

- 素性の定義がとても重要

# nltk.MaxentClassifier

---

- NLTKには各種の機械学習アルゴリズムが含まれている
  - `nltk.classify.*`  
標準化された素性フォーマット, 簡易的な評価が行える
  - 今回は最大エントロピー法分類器 (`nltk.MaxentClassifier`) を用いる
- `MaxentClassifier`は教師あり分類器である
  - 正解ラベルと素性集合が学習データとして必要
  - 学習事例 (Training Examples) はラベルと素性集合の組
  - CLCコーパスは訂正情報付き！正解ラベルはある
  - となると、残るはどのような素性を使うか...

# nlTK.MaxentClassifierのための学習事例

---

- nlTK.classifyが扱える事例のフォーマット

```
training_set =  
[({"feature":value, "feature":value,...}, label),  
 ({ "feature":value, "feature":value,...}, label),  
 ...]
```

- 素性集合: {"feature":value, "feature":value,...}
- 前置詞の正しさにはどんな素性(特徴量)が関係しているか...
  - 前後の文脈(単語)や品詞は何なのか
  - 他にもっと良い素性は無いのだろうか...

# 素性を抽出する関数

---

- ある入力に対する素性を抽出する関数をつくる
  - 文中の前置詞に対して, その前後の数単語をとらえたりそれらの品詞を取得したり...

- 簡単な素性関数(前置詞直後の単語)

```
def successor(sentence_list, pindex):  
    return {"succ_%s"%(sentence_list[pindex + 1]): 1}
```

- 今回は二値素性(on/off)のみを用いる

# 素性抽出関数の例：品詞素性

---

- NLTKには各種POS Taggerが含まれている
- 今回はPennTreeBankで学習したnltk.pos\_tagを使う

- 使用例：

- sentence: str, words: list

```
words = ["The", "cat", "is", ...]  
tagged = nltk.pos_tag(words)  
  
tagged:  
[("The", "DT"), ("cat", "NN"), ...]
```

```
words = nltk.word_tokenize(sentence)  
nltk.pos_tag(words)
```

- 入力は一語単位のリスト(nltk.word\_tokenize(sentence))
- feature\_extractor.py を参考に作ってみよう

# 評価の前に...

---

- 学習データとテストデータはあらかじめ分けておく
  - 「正しい評価」をおこなうにはどのような状態が必要だろうか
  - テストデータが学習データに含まれると何がまずい？
- 評価尺度
  - 正解率 (accuracy) を評価尺度とする

$$\text{accuracy} = \frac{\text{(num. of correct output)}}{\text{(num. of input instances)}}$$



# 分類器の学習・テストのしかた

---

- 学習

```
def train(numexamples = <xxx>, numiter = <xxx>):  
    trainset_features = <a list of dictionaries>  
    trainset_labels = <a list of strings>  
    trainset = zip(trainset_features, trainset_labels)  
    classifier =  
        nltk.MaxentClassifier.train(trainset, algorithm="IIS",  
                                    max_iter=numiter)
```

- テスト

```
def test(classifier):  
    testset_features = <a list of dictionaries>  
    testset_labels = <a list of strings>  
    testset = zip(testset_features, testset_labels)  
    accuracy = nltk.classify.accuracy(classifier, testset)
```

# 休憩？

---

学習には時間がかかります。  
終わらないかも・・・



# 評価：そのモデルで大丈夫？

---

- Accuracyはどうでしたか？
- MaxentClassifierは多クラス分類器
  - 本当は “at” だったのに “in” だと出力した場合と “on” だと出力した場合はどちらが多いんだろう...

分類器が間違いやすいケースを特定したい！

- そこで混同行列(Confusion Matrix)！

```
classifier_outputs =  
    [nltk.MaxentClassifier.classify(t) for t in testset_features]  
cm = nltk.ConfusionMatrix(gold_labels, classifier_outputs)  
print cm
```

# より良いモデルを求めて...

---

- どの素性が精度に寄与しているのか調べたい...

```
print nltk.MaxentClassifier.show_most_informative_features(n = 10)
```

- 上位10個の素性とそのラベルが表示される
- シンプルな素性では正しい文の特徴を捉えられないときがある
  - 誤り事例や, 元のコーパスをもう一度眺めてみる...そして考える
  - 単語の活用形ではなく, 原形を入れてみてはどうだろうか
  - より広い範囲の構文的な情報を取り込むとどうなるだろうか
  - 誤りの傾向は? 素性に取り込むにはどうするか . . .





Congratulations!!

おつかれさまでした！

また会う日まで・・・