
多項分布のパラメータ推定

```
// catList : カテゴリのリスト
var catList []TypeCat

// PredictCat : 与えられた文書 doc のカテゴリを推定する関数。暫定版。
func PredictCat(doc TypeDoc) (cat TypeCat) {
        cat = catList[0]
        maxValue := ProbCatGivenDoc(doc, cat)
        for i:=1; i<len(catList); i++ {
```

```
                result := probCatGivenDoc(doc, catList[i])
                if result > maxValue {
                        maxValue = result
                        cat = catList[i]
                }
        }
        return
}
```

```
// ProbCatGivenDoc : 文書 doc がカテゴリ cat に含まれる確率の比を求める関数。暫定版。
func ProbDocGivenCat(doc TypeDoc, cat TypeCat) float64 {
        return ProbCat(cat) * ProbDocGivenCat(doc, cat)
}
```

```
var numAllDocs int // すべての文書の数
var numDocsCat map[TypeCat]int // 各カテゴリごとの文書の数

// ProbCat : カテゴリ cat の確率＝カテゴリ cat の文書の全文書に対する割合
func ProbCat(cat TypeCat) float64 {
```

```
    return float64(numDocsCat[cat]) / float64(numAllDocs)
}
```

```
// TypeDoc : 文書の型。各単語の出現個数のmap。
type TypeDoc map[TypeWord]int
// 例：doc[word] = 単語 word が文書 doc に含まれる個数

// ProbDocGivenCat : 文書 doc がカテゴリ cat に含まれる確率
func ProbDocGivenCat(doc TypeDoc, cat TypeCat) (r float64) {
        r = 1.0
        for word, num := range doc {
                r *= math.Pow(ProbWordGivenCat(word, cat), float64(num))
        }
        return
}
```

```
// numWordInCat: ある単語があるカテゴリに含まれる個数
var numWordInCat map[TypeCat]map[TypeWord]int
// 例：numWordInCat[cat][word] = 単語 word がカテゴリ cat に含まれる個数

// numAllWordsInCat : カテゴリに含まれる個数
var numAllWordsInCat map[TypeCat]int
// 例：numAllWordsInCat[cat] = カテゴリ cat に含まれる単語の個数

// ProbWordGivenCat : 単語 word がカテゴリ cat に含まれる確率
func ProbWordGivenCat (word TypeWord, cat TypeCat) float64 {
    return float64(numWordInCat[cat][word])/float64(numAllWordsInCat[cat])
}
```

```
var numAllWords int // 全単語数

// ProbWordGivenCat : 単語 word がカテゴリ cat に含まれる確率(スムージング拡張版)
func ProbWordGivenCat (word TypeWord, cat TypeCat) float64 {
    num := float64(numWordInCat[cat][word] + 1)
    sum := float64(numAllWordsInCat + numAllWords)
    return num/sum
}
```

```
// TypeDoc : 文書の型。各単語の出現個数のmap。
type TypeDoc map[TypeWord]int
```

```
// 例：doc[word] = 単語 word が文書 doc に含まれる個数

// LogProbDocGivenCat : 文書 doc がカテゴリ cat に含まれる確率の対数
func LogProbDocGivenCat(doc TypeDoc, cat TypeCat) (r float64) {
        r = 0.0
        for word, num := range doc {
                r += float64(num) * math.Log(ProbWordGivenCat(word, cat))
        }
        return
}
```

```
// LogProbCatGivenDoc : 文書 doc がカテゴリ cat に含まれる確率の比の対数
func LogProbDocGivenCat(doc TypeDoc, cat TypeCat) float64 {
        return math.Log(ProbCat(cat)) + LogProbDocGivenCat(doc, cat)
}
```

```
// PredictCat : 与えられた文書 doc のカテゴリを推定する
func PredictCat(doc TypeDoc) (cat TypeCat) {
        cat = catList[0]
        maxValue := LogProbCatGivenDoc(doc, cat)
        for i:=1; i<len(catList); i++ {
                result := LogProbCatGivenDoc(doc, catList[i])
                if result > maxValue {
                        maxValue = result
                        cat = catList[i]
                }
        }
        return
}
```

```
// wordList : 単語のリスト
var wordList map[TypeWord]int

// Train : 文書 doc をカテゴリ cat として学習する
func Train(doc TypeDoc, cat TypeCat) {

	// カテゴリ cat が初出かどうか検査する
	_, ok := numDocsCat[cat]
	if !ok { // カテゴリ cat が初出の場合
		// カテゴリリストに追加
		catList = append(catList, cat)
		// カテゴリの文書を初期化
		numWordInCat[cat] = map[TypeWord]int{}
	}

	// すべての文書の数をインクリメント
	numAllDocs++

	// カテゴリ cat の文書の数をインクリメント
	numDocsCat[cat] = numDocsCat[cat] + 1

	// 文書 doc に出現する単語 word についてそれぞれ処理
	for word, num := range doc {
		// カテゴリ cat に含まれる単語の個数をインクリメント
		numAllWordsInCat[cat] = numAllWordsInCat[cat] + num
		// 単語 word がカテゴリ cat に含まれる個数をインクリメント
		numWordInCat[cat][word] = numWordInCat[cat][word] + num
		// 単語 word を単語リストに追加。
		wordList[word] = wordList[word] + num // 単語の出現回数
	}
```

```
        // すべての単語の重複のない個数を計算
        numAllWords = len(wordList)
}
```

全体の指標ををまとめた表を以下に示す。

Micro Precision	0.913161
Micro Recall	0.913161
Micro F-Measure	0.913161
Macro Precision	0.913058
Macro Recall	0.914983
Macro F-Measure	0.914019
Overall Accuracy	0.980703

単純な実装内容にも関わらず、高い精度でカテゴリ推定できることがわかった。

参考：Livedoor ニュースコーパス
https://www.rondhuit.com/download.html#ldcc

---

5. おわりに

　本稿では、文書のカテゴリを推定する分類器をナイーブベイズを用いたモデルで示し、そして GoLang による実装例を示した。また、Livedoor ニュースコーパスを用いて評価した結果を示した。

　今後はマルチラベル分類器に拡張していく予定である。