Rekonstrukcja, klasyfikacja, kolokacje

Paweł Rychlikowski

Instytut Informatyki UWr

6 listopada 2018

Przypomnienie zadania z zeszłego wykładu

- Mówiliśmy o przykładowym zadaniu rekonstrukcji samogłoskowej.
- Czyli zamianie:

  stfn pjchł zbczć ltrn w stc _ kłbrzsk cntrm kltr

  na

  stefan pojechał zobaczyć latarnię w ustce i kołobrzeskie centrum kultury

Podejście 1. Powtórzenie

- Model unigramowy
- Dla wyciągu spółgłoskowego bierzemy najczęstsze słowo o tym wyciągu.

Prezentacja

- Popatrzmy jak działa w praktyce model unigramowy.
- Uruchamiamy program:
  `unigramowy_rekonstruktor_samogłoskowy.py`

Wnioski z prezentacji

Z prezentacji wyciągamy następujące wnioski:

- Unigramowy model językowy w zadaniach rekonstrukcji tekstu raczej nie jest ostatecznym rozwiązaniem.
- Z drugiej strony zaskakująco dobrze czasem radzi, również zgadując poprawną formę słowa, względnie poprawne gramatycznie ciągi słów.

Model bigramowy w służbie zadania rekonstrukcji

- Wybieramy ciąg, maksymalizujący (interpolowane) prawdopodobieństwo bigramowo-unigramowe.
- Przestrzeń od poprawy: uwzględnienie 3-gramów, uwzględnienie gramatyki (ew. sufiksów)
- Drobny problem do rozwiązania: decyzje o wyborze słowa **przestają być niezależne**.

Jak to działa?

Inicjacja struktur

```
def best_sequence(codes):
  variants = []
  for c in codes:
    variants.append(normal_forms[c])

  # costs[i][w]: cost of the best sequence
  #  ending at posistion i with the word w
  # prev[i][w]: previous word in the best seq.

  costs = [{}]
  prev  = [{}]
  for v in variants[0]:
    costs[0][v] = log_unigram_score(v)
    prev[0][v] = "<START>"
```

Jak to działa? (2)

Dynamiczne znajdywanie najlepszej sekwencji i kosztów

```
for i in range(1,len(variants)):
  costs.append({})
  prev.append({})

  for w2 in variants[i]:
   candidates = [ (costs[i-1][w1] +
      log_bigram_score(w1,w2), w1)
      for w1 in variants[i-1]]
   c, w = max(candidates)
   costs[-1][w2] = c
   prev[-1][w2] = w
```

Jak to działa? (3)

Odtwarzanie sekwencji

```
val, last_word = max([ (c,w)
            for (w,c) in costs[-1].items()])
result = [last_word]

for i in range(len(L)-1,0,-1):
  result.append( prev[i][result[-1]])

result.reverse()
return result
```

N-gramy wyższych rzędów

- Można zmodyfikować algorytm, żeby uwzględniał nie 2-gramy, lecz 3-gramy.
- Oczywiście kosztem czasu działania i zużycie pamięci, bo kluczem będzie nie ostatnie słowo, lecz para słów: ostatnie i przedostatnie słowo.

Inny wariant

Można pamiętać nie najlepszy wynik, ale k najlepszych wyników (teoretycznie może to służyć do losowania na przykład tekstu o zadanych sufiksach)

Przykładowe wyniki

INPUT: babuleńka miała dwa rogate koziołki
REPRESENTATIONS: bblńk mł dw rgt kzłk
UNIGRAM: babuleńka miał dwa regaty koziołek
BIGRAM: babuleńka miał dwa rogate koziołek

INPUT: judyta podarowała wczoraj stefanowi czekoladki
REPRESENTATIONS: jdt pdrwł wczrj stfnw czkldk
UNIGRAM: judyta podarował wczoraj stefanowi czekoladki
BIGRAM: judyta podarowała wczoraj stefanowi czekoladki

Przykładowe wyniki

INPUT: zaśmiał się demonicznie , owinął peleryną i zniknął we mgle
REPRESENTATIONS: zśmł s dmnczn , wnł plrn _ znknł w mgl
UNIGRAM: zaśmiał się demoniczny , owinął palarni i zniknął w mogli
BIGRAM: zaśmiał się demonicznie , owinął pelerynę i zniknął we mgle

INPUT: wiktor wprost uwielbiał jeździć tramwajem bez celu
REPRESENTATIONS: wktr wprst wlbł jździć trmwjm bz cl
UNIGRAM: wiktor wprost uwielbiał jeździć tramwajem bez celu
BIGRAM: wiktorii wprost uwielbiał jeździć tramwajem bez celu

dalsze zastosowania

- rekonstrukcja ogonkow (to juz calkiem uzyteczne zadanie)
- mozliwosci roznego laczenia modeli
- bedzie zadanie na pracownie, polaczone z rekonstrukcja malych i duzych liter

Dalsze zastosowania

- Rekonstrukcja ogonków (to juz całkiem użyteczne zadanie)
- Możliwości różnego łączenia modeli
- Będzie zadanie na pracownię, połączone z rekonstrukcją małych i dużych liter

Czy rekonstrukcja ogonków jest trudna?

- Wydaje się, że wiele słów ma jednoznaczną formę polskawą: mozliwosc, czytajacy, sledzic, ...
- Ale są problemy, na przykład:
  a) Związane z gramatyką: czy jednoznaczna to jednoznaczna czy też jednoznaczną (podobnie np. pisze)
  b) Nieliczne słowa są niejednoznaczne, na przykład: kat i kąt
  c) Czasem jednocześnie decydujemy o słowie i jego odmianie: latka, łatka, łatką

Zadania klasyfikacji

- Zadanie klasyfikacji jest klasycznym zadaniem uczenia maszynowego
- Jest ono zdefiniowane następująco:
  - Mamy dany ciąg par $\{(\mathbf{x}_i, y_i)\}_i$, gdzie $\mathbf{x}_i$ jest wektorem cech, a y_i elementem niewielkiego zbioru klas. Cechami mogą być liczby, wartości logiczne, napisy, ...
  - Szukamy algorytmu, który będzie przewidywał wartości y dla danych $\mathbf{x}$
  - Powinien radzić sobie dobrze dla nieznanych wartości $\mathbf{x}$

Przykładowe zadania klasyfikacji

1. Ustalanie autorstwa – nie tylko książki, ale np. postu (detekcja podwójnych tożsamości)
2. Detekcja spamu – dostępne też inne cechy oprócz gołego tekstu, na przykład temat czy nadawca
3. Detekcja wydźwięku – czy recenzja jest pozytywna, czy negatywna
4. Znajdywanie języka – jaki jest język tekstu? (tu pomaga zejście do poziomu literek, nie tylko wyrazów)
5. Tematyka tekstu – użyteczna na przykład w segregatorach newsów

Przykładowe zadania klasyfikacji. Mniej oczywiste przykłady

1. znaczenie kropki – czy kropka jest końcem zdania, końcem skrótu wewnątrzzdaniowego, końcem liczebnika, elementem bytu informatycznego (takiego jak URL, czy IP)
2. czy słowo play jest rzeczownikiem, czy czasownikiem?
3. czy opinia jest wiarygodna?
4. jaką ocenę powinien dostać autor eseju?
5. czy warszawa powinna być napisane wielką literą?
6. czy mgr. jest błędem?

Naiwny klasyfikator Bayesowski

- Oparty na rachunku pradwopodobieństa i twierdzeniu **Bayesa**,
- na cechach, które uznajemy za istotne
- oraz na **naiwnym** (i na ogół fałszywym) założeniu, że te cechy są niezależne
- niemniej jednak w wielu przypadkach działa i stanowi dobry punkt startowy

Naiwny klasyfikator Bayesowski. Wyprowadzenie

Interesuje nas

$$c = \text{argmax}_{c \in C} P(c|x_1, \ldots, x_n)$$

gdzie c jest wybraną klasą, C zbiorem klas, a x_i to zaobserwowane cechy dla konkretnego przypadku zadania klasyfikacji.
Korzystamy z twierdzenia Bayesa:

$$P(c|x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n|c)P(c)}{P(x_1, \ldots, x_n)}$$

Mianownik jest niezależny od klasyfikacji, zatem (zakładając **naiwnie** niezależność) wybieramy zgodnie z wzorem:

$$c = \text{argmax}_{c \in C} P(c) * \prod_i P(x_i|c)$$

Szacowanie prawdopodobieńst

- Prawdopodobieństwo a-priori, $P(c)$ szacujemy licząc liczbę klasyfikacji równą c podzieloną przez liczbę zadań
- Prawdopodobieństwo a posteriori

$$P(x_i|c) = \frac{P(x_i, c)}{P(c)} = \frac{\text{cnt}(x_i, c)}{\text{cnt}(c)}$$

NB a LM

- Gdy cechami są po prostu słowa (na pozycjach) wówczas wzór wygląda tak:

$$c = \text{argmax}_{c \in C} P(c) * \prod_{i \in \text{Positions}} P(w_i|c)$$

- Czyli NB jest modelem unigramowym z dodatkowym priorem (pozwalającym odróżnić Remigiusza Mroza od Wisławy Szymborskiej)

Wygładzanie

Oczywiście jest konieczne. Często wystarcza podstawowe wygładzanie Laplace'a (czyli +1)

Sentiment analyssis

- **ustalanie wydźwięku**
- Dość oczywiste znaczenie wobec mnóstwa opinii w Internecie
- Na ogół myślimy o nim jako o klasyfikacji tekstu (tweet, dokument, ...), ale współcześnie czasem myśli się o mniejszych kawałkach

Przykład

Stefan powiedział, że „Koziołek Matołek" to straszliwy gniot, a nie piękna literatura.

- Frazy pozytywne: piękna; piękna literatura
- Frazy negatywne: Matołek; straszliwy gniot; straszliwy gniot, a nie piękna literatura
- Frazy neutralne: Koziołek Matołek; Stefan powiedział (...) litaratura., i inne

Dwumianowy NB

- W przypadku klasyfikacji wydźwięku interesuje nas często sam fakt występowania słowa (super, rewelacja, klapa, ...), a nie liczba tych wystąpień.
- W takiej sytuacji stosujemy dwumianowy NB (binomial), a cechą jest po prostu: w klasyfikowanym tekście wystąpiło X.
- Pozostał do rozwiązania pewien ważny problem. Jaki?

Jest różnica pomiędzy: To **nie** jest fajny film a To jest fajny film

Negacja

Rozwiązuje się to (w pierwszym przybliżeniu) bardzo łatwo: dla każdego słowa mamy dwie cechy:

a) Słowo pojawiło się, a przed w okienku ok. 5 słów było słowo negujące (nie, żaden, ...)

b) Słowo pojawiło się, bez wyrażenia negującego

Klasyfikacja SPAMU

- Można dodawać inne cechy, niekoniecznie słowa (na przykład Orzeszkowa być może lubiła długie zdania, więc cecha: zdanie ma więcej niż 20 wyrazów może być przydatna w klasyfikatorze P-O-S.
- Jakie cechy do klasyfikacji SPAM-u:
  1. Temat napisany WIELKIMI LITERAMI
  2. Zawiera frazy ponaglające, takie jak : urgent reply
  3. Temat zawiera online pharmaceutical
  4. Błędy w HTML-u (niezbalansowane tagi)

Klasyfikacja języka

- Lepiej sprawdzają się cechy postaci: tekst zawiera **s**, gdzie **s** jest krótkim ciągiem literek (powiedzmy od 1 do 4).
- Zobacz: `https://github.com/saffsd/langid.py`
- 97 języków, około 7000 cech (wybranych).

Krótkie demo: `import langid`

Jeszcze trochę o modelowaniu języka

Problem

Jak modelować odległe zależności (wykraczające poza N, gdzie N oznacza sensowne granice N-gramowości)

Idea 1

Dorzucamy czynnik: $P(w|\text{klasa-dokumentu})$ (oczywiście możemy wyznaczyć tę klasę wcześniej, na przykład za pomocą NB).

- Jak wiemy, że dokument jest chemiczny, to możemy „boostować” p-stwo takich słów jak: azotyn, tlenek, spalanie, karbamid, ...
- To że dokument jest chemiczny, może wynikać z bardzo odległych słów...

Jeszcze trochę o modelowaniu języka (2)

Idea 2

Cache – dodajemy do prawdopodobieństwa unigramowego czynnik związany z historią (dużo dłuższą niż N), zwiększający prawdopodobieństwo słów ostatnio widzianych.

- Trochę niebezpieczne jeżeli tekst sami sobie rekonstruujemy, bo może wzmacniać błędy...

Idea 3

Skip-gramy – czyli dorzucenie prawdopodobieństwa bigramowego z przeskokiem, czyli $P(w_i|w_{i-k})$

- można też grupować k i rozważać $P(w|\text{w oknie o długości } k \text{ wcześniej było } w')$

Kolokacje

- Pewne bigramy występują obok siebie przypadkowo, inne tworzą ciekawe połączenia.
- Jak odróżnić jedne od drugich?
- Możliwe dwa podejścia (jakie?):
  - Bazujące na gramatyce (tworzą sensowną frazę, na przykład rzeczownik i przymiotnik)
  - Bazujące na statystyce (analizujemy częstości składników i samego bigramu)

Zadanie

Będziemy testować kolokacje na przykładzie następujących słów

- dziewczyna, kobieta
- helekopter, śmigłowiec
- chłopak, mężczyzna
- herbata, kawa
- piwo, wino

Będziemy bazować na bigramach, zatem kolokacją będzie para słów, w których jedno jest przez nas wybrane.

Kolokacje (2)

Jak najprościej wybrać kolokacje?
Metodą zerową jest wybieranie po prostu najczęstszych bigramów.

Podejście 1

Wybieramy najczęstsze bigramy dla słów.

Kobieta

8910 kobieta ,
7033 kobieta .
5701 . kobieta
1377 kobieta w
1293 , kobieta
1193 że kobieta
1190 kobieta -

Mężczyzna

7174 mężczyzna ,
6225 . mężczyzna
5364 mężczyzna .
1460 mężczyzna w
1244 letni mężczyzna
1083 że mężczyzna

Podejście 1 (cd)

Helikopter

220 helikopter .
168 helikopter ,
80 . helikopter
40 helikopter z
34 , helikopter
32 helikopter w
30 ”helikopter

Śmigłowiec

. śmigłowiec 334
, śmigłowiec 224
śmigłowiec . 118
z śmigłowiec 51
śmigłowiec , 33
mi śmigłowiec 30
”śmigłowiec 30

Podejście statystyczne

- Interesują nas słowa, które się „mają ku sobie”.
- Pewne bardzo częste słowa moga występować w swojej okolicy „przypadkowo”, a zatem interesuje nas:
  1. Bigram jest możliwie częsty
  2. Pojedyncze słowa są możliwie rzadkie

Przykłady (NKJP)

- Częste słowa: w,i,z,pan,poseł
- Popatrzmy na częstości bigramów oraz na hipotetyczne częstości bigramów, przy założeniu, że $P(w_1 w_2) = P(w_1)P(w_2)$:

```
i w   = 103593    (oczekiwana) 166K)
w i   = 6410      (oczekiwana jw.)
z na  = 147       (oczekiwana 71K)
na z  = 153
pan poseł = 20884 (oczekiwana 45)
poseł pan = 6
```

Cóż, nie wygląda to całkiem idealnie, ale...

Pointwise Mutual Information

- Inną opcją jest PMI
- Sortujemy wg

$$\log\left(\frac{P(w_1 w_2)}{P(w_1)P(w_2)}\right)$$

Positive Pointwise Mutual Information jest równe $\max(0, PMI(w_1, w_2))$

- czyli jak coś jest rzadziej niż „przypadkowo” to dajemy 0