Gépi tanulás – áttekintés

A számítógépes nyelvészet alapjai – 2022/23 tavasz
7. óra

Simon Eszter

2023. április 24.

Tartalom

Racionalista és empirikus megközelítés

Racionalista és empirikus megközelítés

Racionalista:

- szabályalapú
- a nyelvész írja a szabályokat $\rightarrow$ a nyelvi információt expliciten adja át a számítógépnek

Empirikus:

- statisztikai alapú
- a számítógépes nyelvész az erőforrásokat adja oda a számítógépnek, ami azokat felhasználva magát tanítja

Kérdés: amikor számítógépes nyelvmodellt építünk, akkor korpuszadatokra vagy introspekcióra támaszkodunk?

A szabályalapú rendszerek előnyei és hátrányai

Előnyei:

- a fejlesztő nagyobb kontrollja a rendszer felett
- könnyebben értelmezhető visszacsatolás a rendszertől
- magas pontosság

Hátrányai:

- sok kézi munkát és nagy szakértelmet kíván
- nem hibatűrő
- bonyolult a fejlesztése, törékeny
- nehezen vihető át más nyelvre/doménre

A statisztikai alapú rendszerek előnyei és hátrányai

Előnyei:

- minden elemzéshez egy valószínűséget kapunk → rangsorolhatjuk őket → kiválaszthatjuk a leginkább odaillőt
- még akkor is adhat jó eredményt, ha a mögöttes nyelvmodell nem adekvát
- sokkal flexibilisebb megközelítés

Hátrányai:

- nagy mennyiségű annotált adatot igényelnek → a kézi munka nem tűnt el, csak átalakult
- a rendszer átvitele más nyelvre/doménre elég nagy teljesítménybeli visszaesést okoz

A statisztikai módszerek kategorizálása

- felügyelt (supervised)
- félig felügyelt (semi-supervised)
- felügyelet nélküli (unsupervised)

Felügyelet nélküli és félig felügyelt tanulás

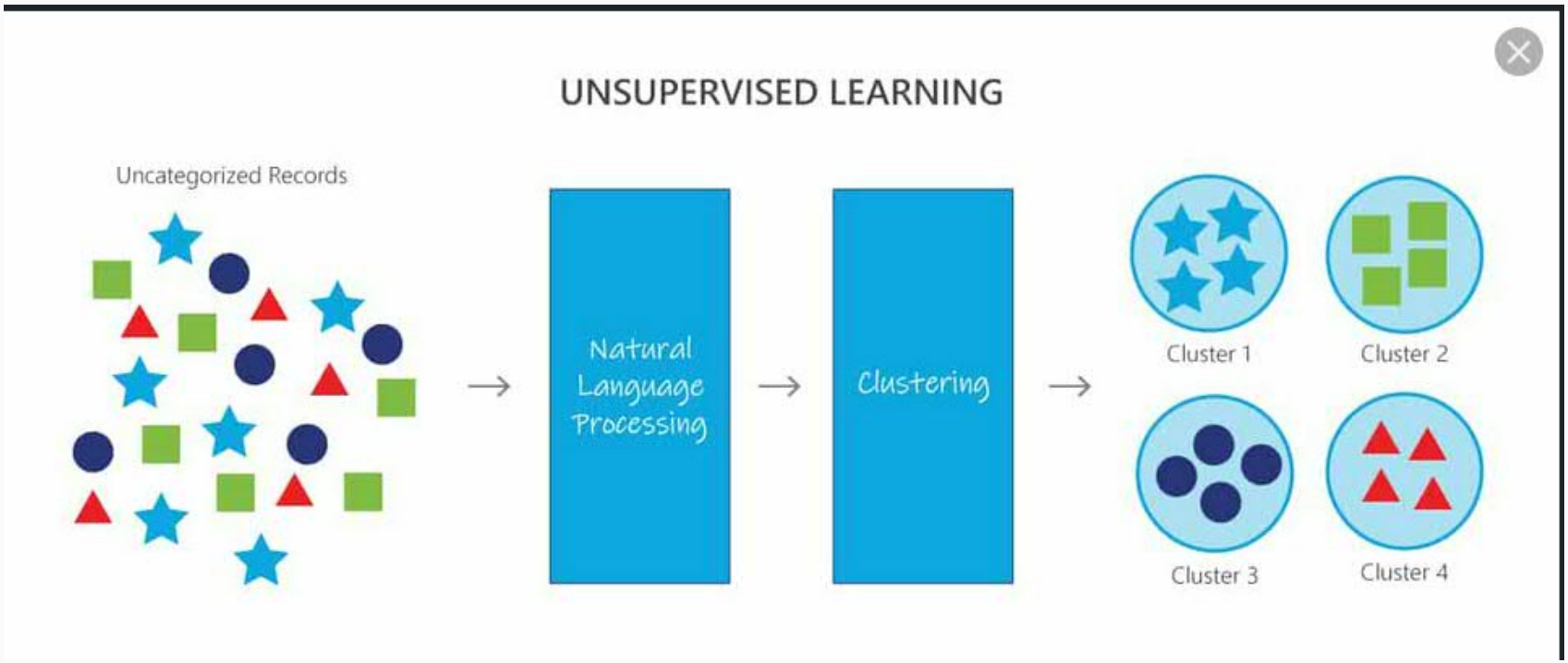
Felügyelet nélküli tanulás

előfeldolgozott szöveg címkék nélkül → kérdés: mit lehet megtanulni a nyers szövegből?

Klaszterezés:
a hasonló grammatikai viselkedésű elemek fognak osztályokba csoportosulni

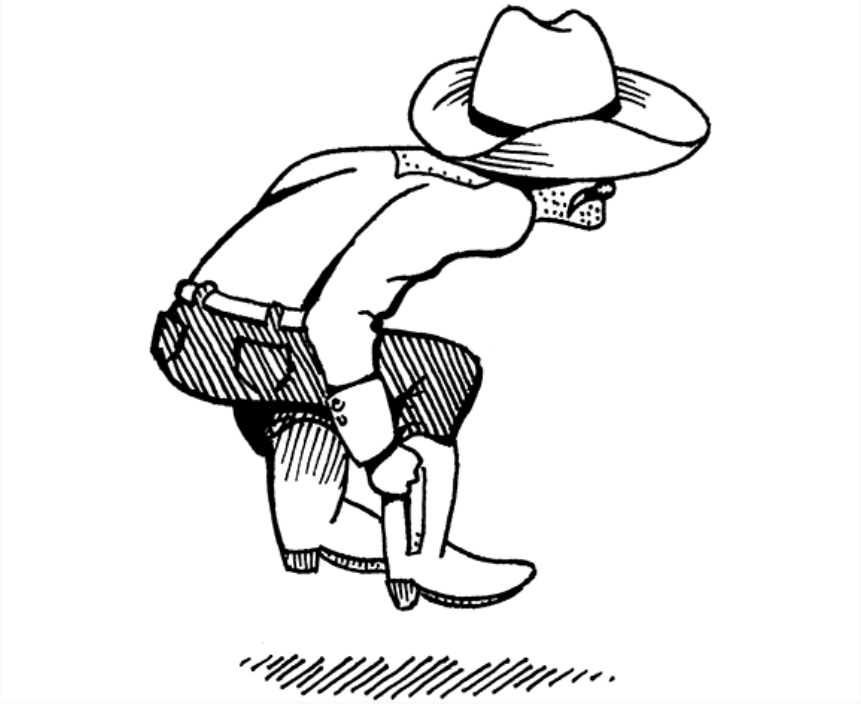
1. nincsenek előre definiált osztályok
   - ha új típusokat akarunk találni
2. előre megszabjuk az osztályok számát
   - ha az adott feladatban megszokott osztályok szerint akarjuk kiértékelni

A klaszterezés



Félig felügyelt tanulás

- címkézetlen szövegből tanul
- kézzel összeállított kiinduló halmaz (’seed’)
- bootstrapping
- az adatban előforduló természetes redundanciára építenek



A felügyelt gépi tanulás menete

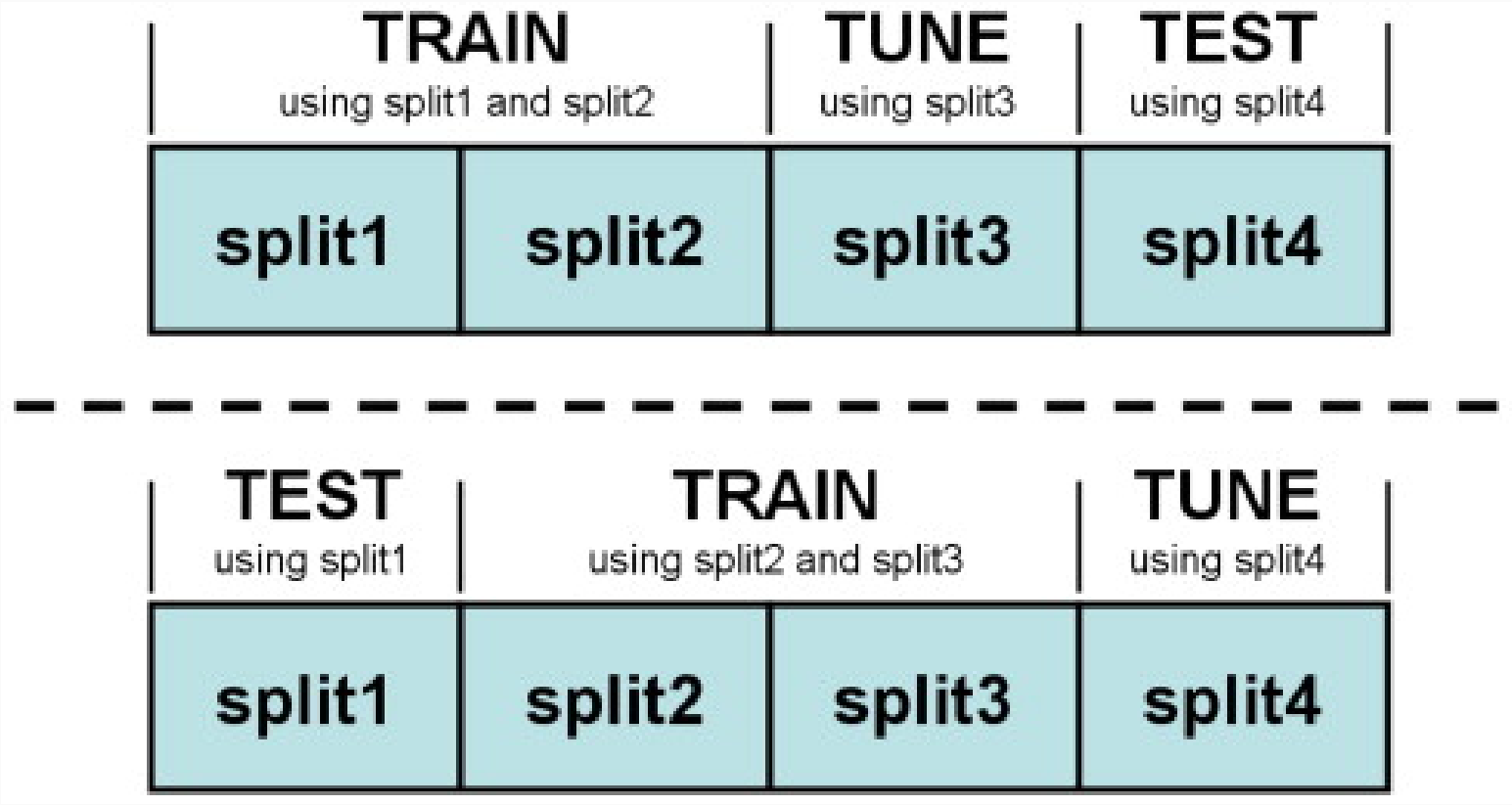
A felügyelt gépi tanulás alapvetése

- a felügyelt gépi tanulás azon a feltételezésen alapul, hogy az adatpontok egymástól független elemek, amelyeknek egyenletes az eloszlásuk
- feltesszük, hogy az eddig nem látott adatpontokra is igaz ez $\rightarrow$ így tudunk következtetni a már látott nyelvi elemekből a még nem látottakra
- nyelvi annotációval ellátott korpusz $\rightarrow$ ebből tanulja ki az adott adatpontokra jellemző jegyeket, és ez szolgál majd a kiértékelés alapjául is

Általános séma

1. gold standard korpusz
2. train–devel–test halmaz, keresztvalidáció
3. jegykinyerés
4. modellépítés
5. taggelés
6. kiértékelés

Train–devel–test & keresztvalidáció



Túltanulás (overfitting)

- a teszthalmaz elemei nem szerepelhetnek a tanító halmazban
- fejlesztés közben mérni csak a develen szabad
- ha nem így teszünk, akkor az nem csak csalás, de a rendszerünk nem lesz képes általánosításokra $\rightarrow$ túlzottan rátanul az adott szövegre
- a rendszer a tanító halmazban levő random zajt reprezentálja, nem általánosít

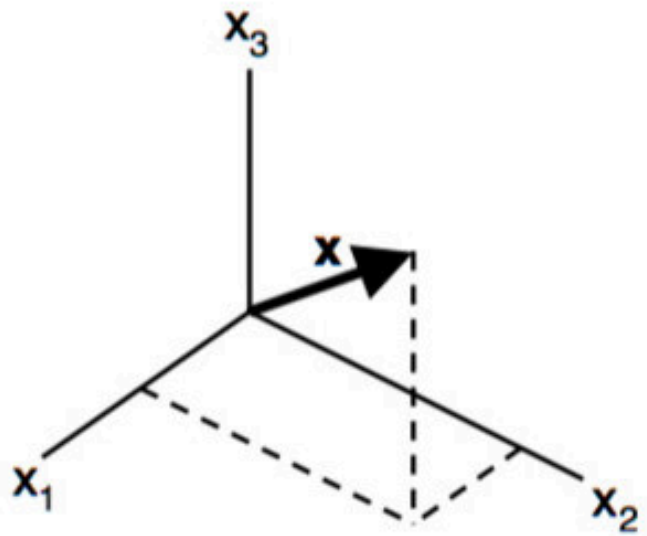
Jegykinyerés (feature extraction)

- a jegyek az adatpontok különféle tulajdonságait írják le
- a jegyeket a számítógépes nyelvész találja ki, definiálja és kódolja
- a jegy hasznosságát az adat határozza meg: a jegy megkülönböztető erejét ki kell mérni, utána eldönteni, hogy alkalmazzuk-e
- a jegyek hozzáadása vagy a paraméterek állítása egyesével, majd mérés, ha nem ront, akkor meghagyjuk
- a jegyek vektorokra képeződnek le

Jegykinyerés (feature extraction)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_d \end{bmatrix}$$

Feature vector



Feature space (3D)

Jegytípusok (HunTag)

forrás: felszíni tulajdonságok, morfológiai információk, szintaktikai információk, listatagság

érték: sztring, bináris

egység: token, mondat

pl. bináris felszíni jegyek: *hascap, allcaps, capperiod, camel, 3caps, iscap*

pl. morfológiai jegyek: *lemma, fulltag, pos, tagend, oov, tagpattern, isbetweensamecases, plural*

pl. szintaktikai jegyek: *sentstart, sentend, NpPart, parsePatts*

pl. listatagság: is-a reláció, pl. Budapest benne van a városok listájában, akkor város

További lépések

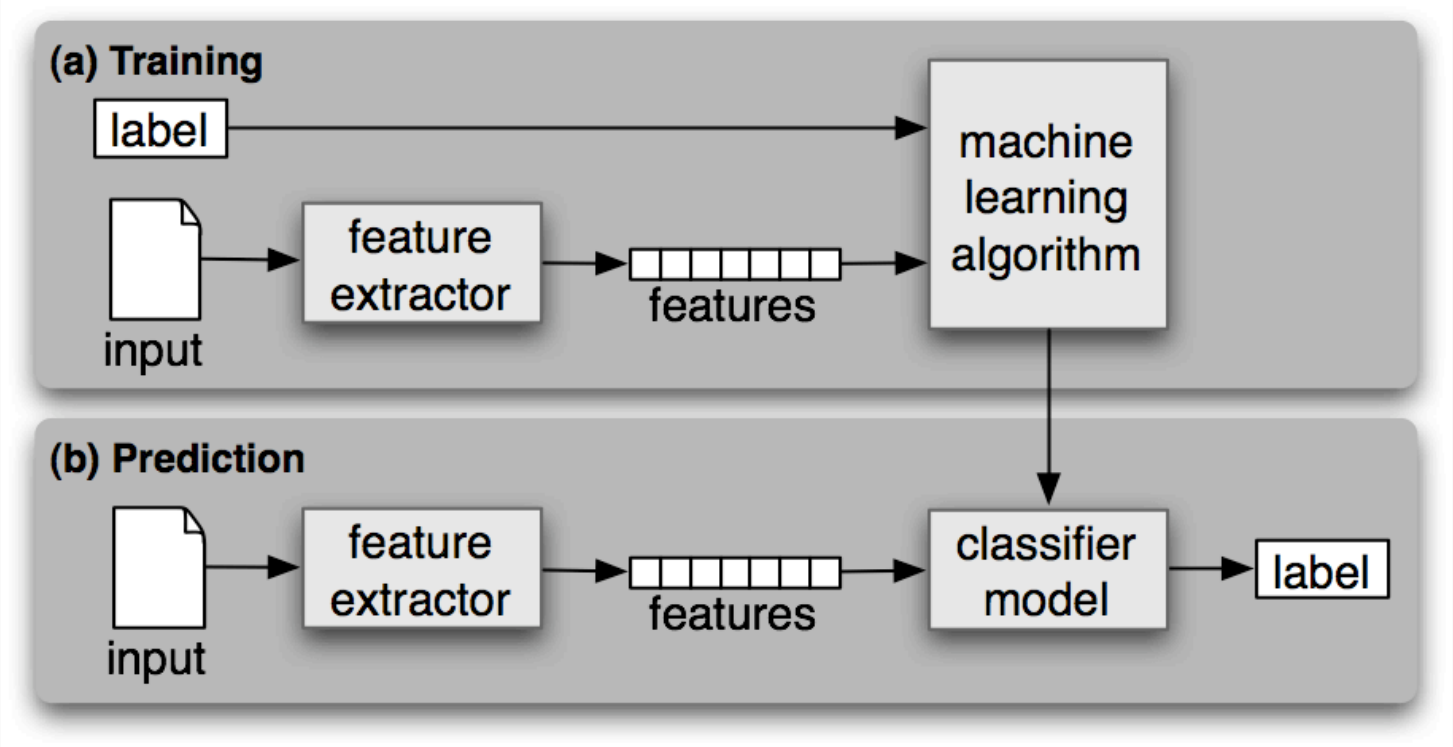
modellépítés:
jegy–címke párok, mindegyikhez egy súly hozzárendelve, ami azt mutatja meg, hogy az adott jegy mennyire van hatással arra, hogy az adott jeggyel rendelkező token az adott címkét kapja

taggelés:

1. a kiértékelő halmaz fícsörizálása
2. a kapott fícsörvektorok és a nyelvmodell alapján címkék kibocsátása

baseline:
a rendszertől minimálisan elvárt teljesítmény $\rightarrow$ jellemzően a leggyakoribb címke kiosztása minden adatpontra

Áttekintés


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

Címkézési feladat

Szekvenciális vs. tokenalapú címkézési feladat

szekvenciális
lánc-struktúra → időbeli vagy térbeli egymásutániság (pl. szavak egy mondatban) → a szekvencia minden egyes eleméhez címkét kell rendelni

tokenalapú
egy adatponthoz (egy tokenhez) egy címkét kell rendelni

Klasszifikáció vs. regresszió

klasszifikáció
kategóriacímkét bocsát ki (két vagy több diszkrét kategória)

regresszió
valós számot bocsát ki (folytonos)

logisztikus regresszió
valós számot bocsát ki, ami valószínűségként értelmezhető (két kategória)

multinomiális logisztikus regresszió
valós számot bocsát ki, ami valószínűségként értelmezhető (több kategória)

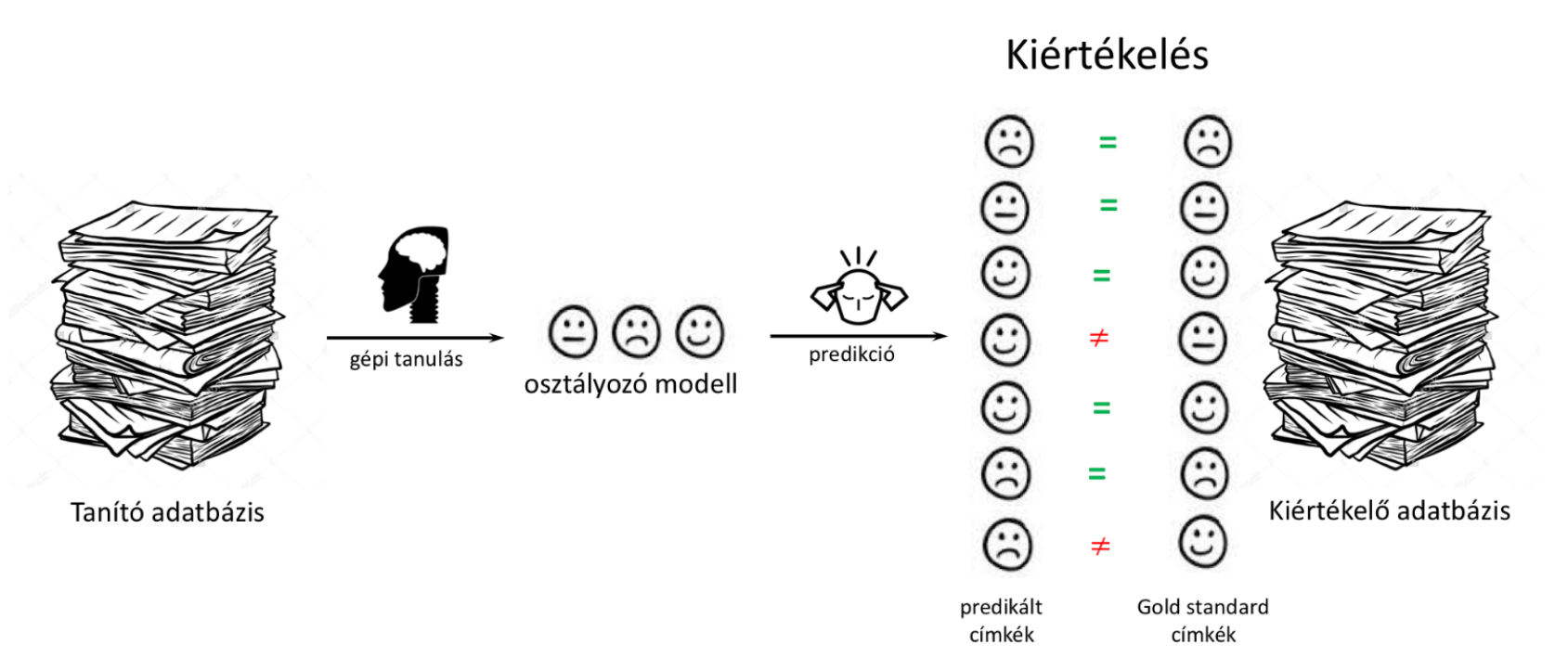
Maximum entrópia

- multinomiális logisztikus regresszió
- tokenalapú
- címkék fölötti teljes valószínűség-eloszlást bocsát ki

token	gold	C1	P1	C2	P2
tájékoztatta	O	O	1		
a	O	O	1		
Magyar	B-ORG	B-ORG	0.92	1-ORG	0.08
Nemzeti	I-ORG	I-ORG	0.96	O	0.04
Bank	E-ORG	E-ORG	1		
csütörtökön	O	O	1		
az	O	O	1		
MTI-t	1-ORG	B-ORG	0.35	1-ORG	0.65

Kiértékelés

Honnan tudhatjuk, hogy egy erőforrás vagy eszköz jó?



Accuracy: az összes predikció közül hány esetben értett egyet a program a kiértékelő adatbázisban szereplő címkével?

Képeket mutatunk, a program jelzi, ha macskát lát.

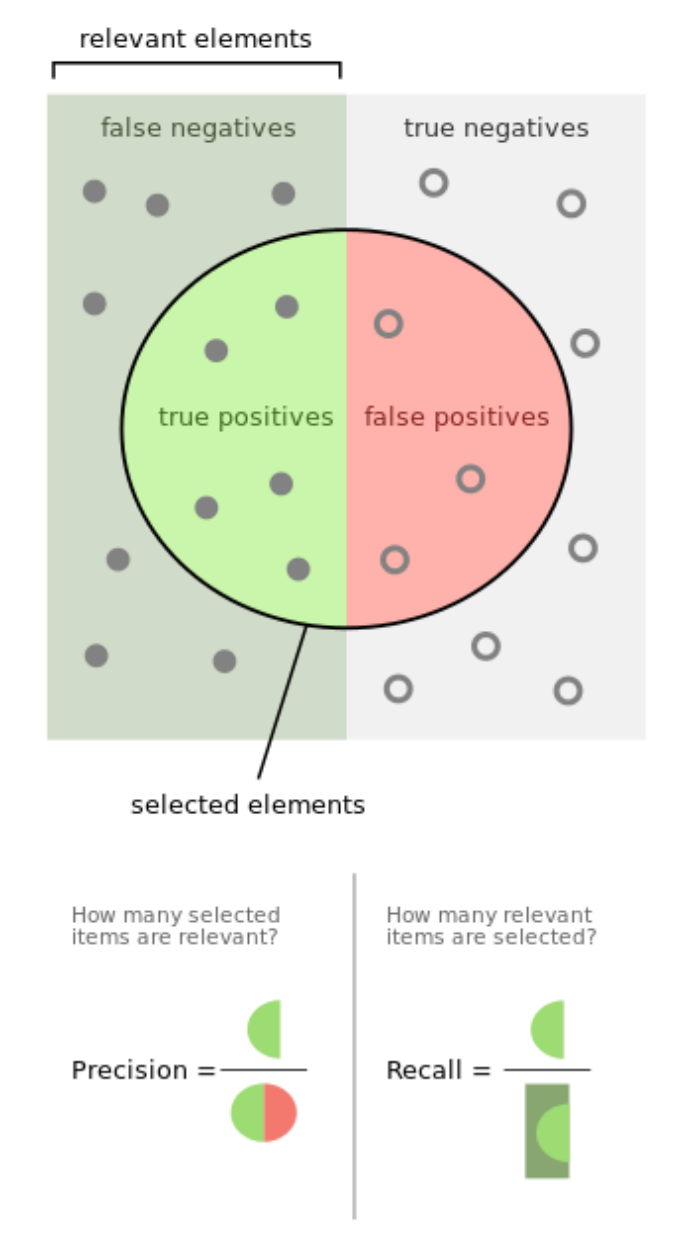
- TP: hit
  macskát mutattunk, macskát mondott
- FN: type II errror, miss, underestimation
  macskát mutattunk, nem mondott semmit
- FP: type I error, false alarm, overestimation
  kutyát mutattunk, macskát mondott
- TN: correct rejection
  kutyát mutattunk, nem mondott semmit

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

Accuracy

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Pontosság és fedés



A “jóság” mérőszámai

Hogy torzíthatjuk az eredményeket?

- a **pontosság** (precision, ami azt mutatja, hogy a találatokból hány volt eredetileg jó) növelésére: a program sose mond semmit
- a **fedés** (recall, ami azt mutatja, hogy az eredetileg jók közül hányat találtunk meg) növelésére: a program mindig macskát mond

Ellenszer: **F-mérték** (F-measure, F-score): a pontosság és a fedés harmonikus közepe, átlaga

Pontosság, fedés, F-mérték

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 \times precision + recall}$$

F-mérték

- a *pontosság* maximalizálása: minél kevesebb tévedés $\rightarrow$ szigorítás
- a *fedés* maximalizálása: minél több találat $\rightarrow$ megengedőbb rendszer

$\beta = 1$

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$