

Khai thác ngũ liệu văn bản và UD

Biểu diễn và phân loại văn bản

Nguyễn Trường Sơn
ntson@fit.hcmus.edu.vn



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung

- Giới thiệu
- Các phương pháp cho bài toán phân loại văn bản
 - Sử dụng luật
 - Sử dụng các cách tiếp cận máy học truyền thống
 - Biểu diễn văn bản
 - Xây dựng mô hình huấn luyện
- Cách đánh giá một hệ thống phân loại văn bản
 - Accuracy
 - Precision / Recall / F1
 - Cross Validation
- Các phương pháp tinh chỉnh tham số: Hyper parameter tuning
- Các phương pháp phân loại sử dụng học sâu

Phân lớp văn bản

- Phân lớp văn bản = text categorization / document classification
 - Mục tiêu: Phân loại một văn bản vào một/một vài loại nào đó đã cho trước



- Một số ứng dụng phổ biến:
 - Phân loại nội dung email thành 1 trong 2 loại thư rác / không
 - Phân loại tin tức vào các chủ đề tin tức: tin thể thao, tin thời sự...
 - Phân loại câu hỏi trong hệ thống hỏi đáp
 - Phân loại ý định người dùng trong các hệ hội thoại (chatbot)
 - “Tôi muốn đặt 2 phòng đơn” → book_a_room
 - “Tôi muốn huỷ đặt phòng”. → cancel_booking
 - Phân loại cảm xúc: tích cực / tiêu cực về sản phẩm, dịch vụ
 - Phân loại bệnh án điện tử.

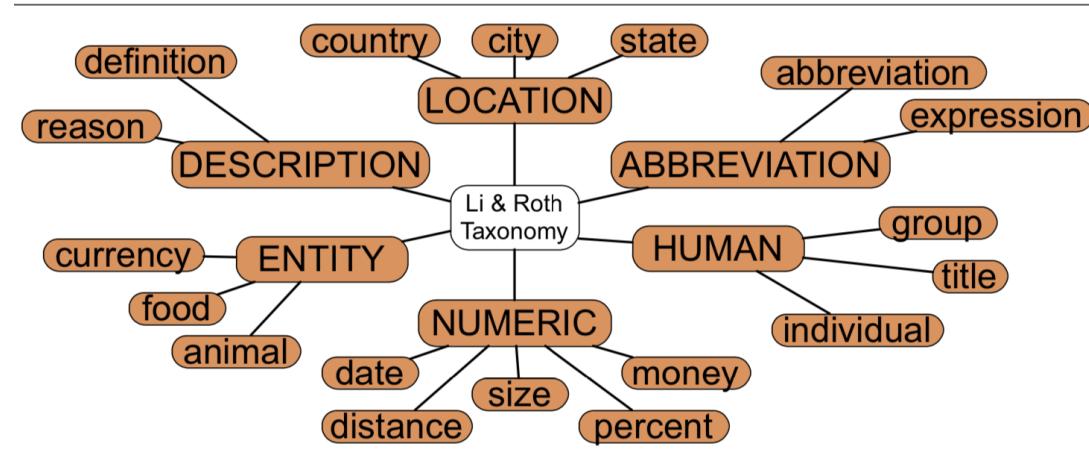
Ví dụ

- **Bài toán phân loại câu hỏi:**
 - Phân loại một câu hỏi về một loại nào đó:
 - Dân số Hà Nội là bao nhiêu → NUM
 - Ai là người đầu tiên lên mặt trăng → HUMAN
 - Nước nào có dân số đông nhất → LOCATION
 - ...
 - Là một bài toán điển hình trong phân loại văn bản
 - Ứng dụng cho các hệ thống hỏi đáp



Ví dụ

- Phân loại câu hỏi:
 - **Bài toán xác định loại câu trả lời**
 - Bài toán 1: Phân loại câu hỏi vào 1 trong 6 loại chính
 - Bài toán 2: Phân loại câu hỏi vào 1 trong 50 loại nhỏ (Li & Roth)



- Nước nào có dân số đông nhất:
 - Mức thô: LOCATION
 - Mức mịn: LOCATION.country
- Kỹ thuật: rule-based, machine-learning, hybrid

Ví dụ

□ TREC 15 Dataset

```
DESC:manner How did serfdom develop in and then leave Russia ?  
ENTY:cremat What films featured the character Popeye Doyle ?  
DESC:manner How can I find a list of celebrities ' real names ?  
ENTY:animal What fowl grabs the spotlight after the Chinese Year of the Monkey ?  
ABBR:exp What is the full form of .com ?  
HUM:ind What contemptible scoundrel stole the cork from my lunch ?  
HUM:gr What team did baseball 's St. Louis Browns become ?  
HUM:title What is the oldest profession ?  
DESC:def What are liver enzymes ?  
HUM:ind Name the scar-faced bounty hunter of The Old West .  
NUM:date When was Ozzy Osbourne born ?  
DESC:reason Why do heavier objects travel downhill faster ?  
HUM:ind Who was The Pride of the Yankees ?  
HUM:ind Who killed Gandhi ?
```

<https://cogcomp.seas.upenn.edu/Data/QA/QC/>



Ví dụ

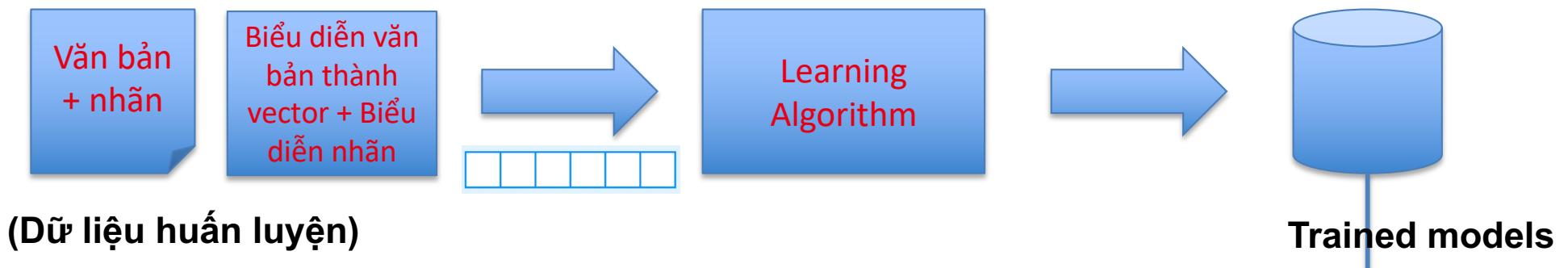
- Phân loại cảm xúc (sentiment analysis)
 - "The hotel is really beautiful. Very nice and helpful service at the front desk." → POSITIVE
 - We had problems to get the Wi-Fi working. The pool area was occupied with young party animals, so the area wasn't fun for us → NEGATIVE
- Số lớp trong bài toán phân loại cảm xúc thường là: 2, 3, 5



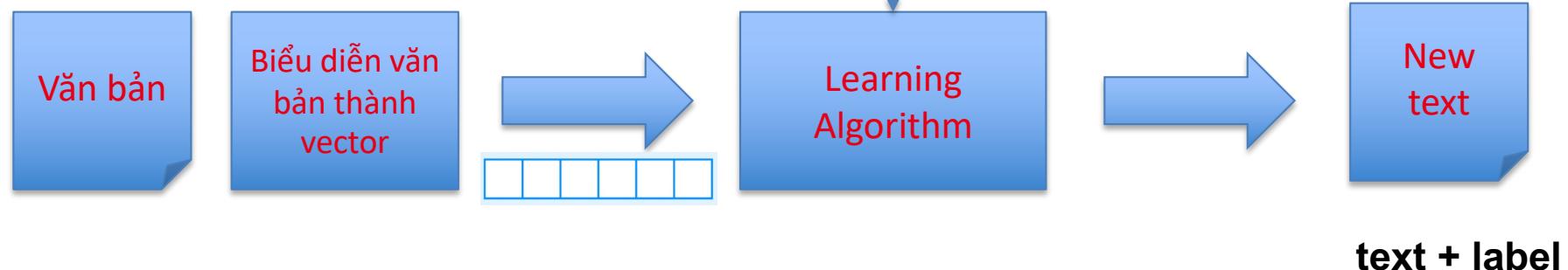
Phân loại văn bản bằng học máy

- Gồm 2 giai đoạn chính:

- Giai đoạn 1: Huấn luyện mô hình:

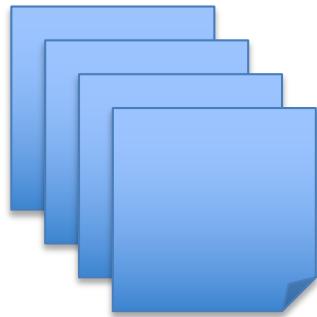


- Giai đoạn 2: Sử dụng mô hình để đoán dữ liệu mới:



Biểu diễn văn bản thành vector đặc trưng

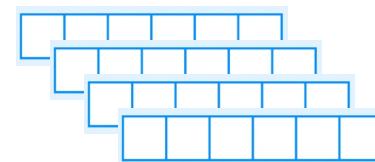
- Văn bản là một chuỗi / mảng (sequence) mảng các:
 - Ký tự
 - **Từ → được sử dụng khá phổ biến**
 - Phù hợp
 - Cụm từ, Câu, Đoạn văn



Văn bản

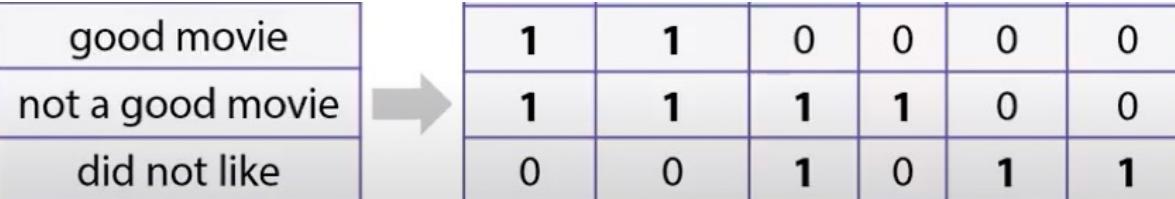
Phương pháp:

- Bag of words
- Deep learning (Advanced ...)



Các vector có độ dài
bằng nhau

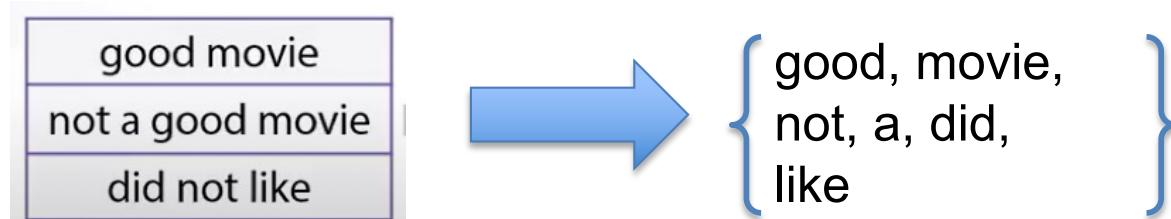
Kỹ thuật: Tách từ , Tách câu, Chuẩn hoá
(dấu câu, hoa thường, ...); Loại bỏ những
từ không có ý nghĩa



good movie	1	1	0	0	0	0
not a good movie	1	1	1	1	0	0
did not like	0	0	1	0	1	1

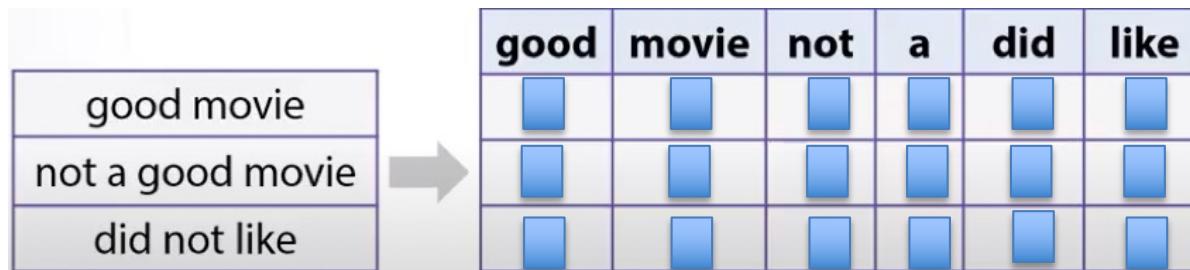
Biểu diễn văn bản thành vector đặc trưng

- Biểu diễn văn bản thành vector bằng phương pháp bag-of-words:
 - Bag of words = túi từ = một tập hợp các **từ/ngữ (cụm từ)** trong kho ngữ liệu
 - Bước 1: Xác định túi từ



vocab , vocab size = 6

- Bước 2: Biểu diễn



- ✓ Độ dài của mỗi vector = vocab size
- ✓ Giá trị: Tuỳ theo phương pháp biểu diễn

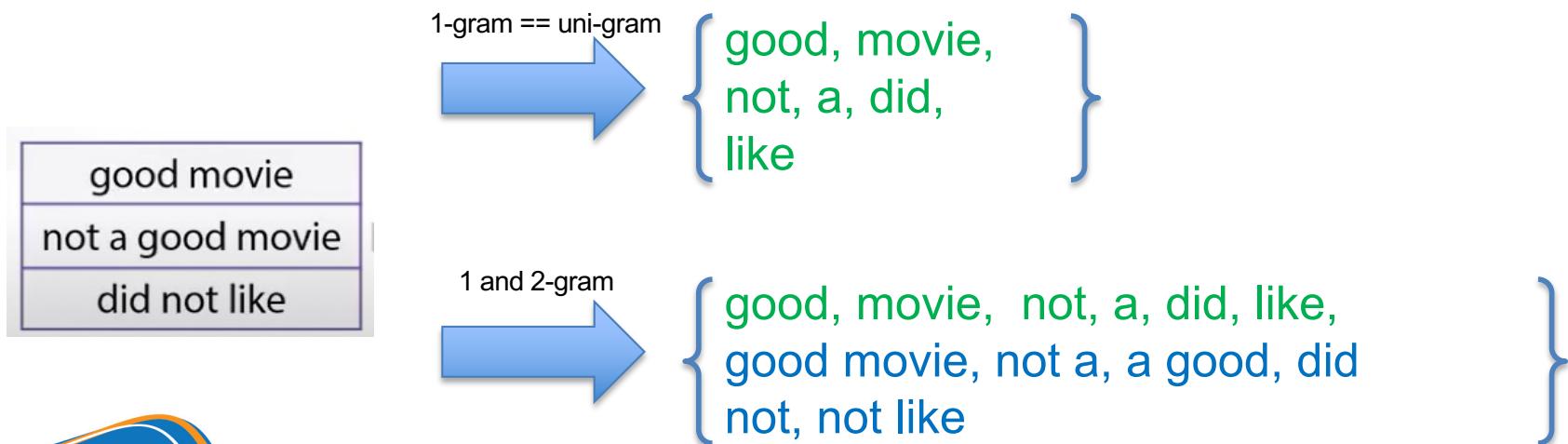
Biểu diễn văn bản thành vector đặc trưng

- Bag-of-words:

- Hạn chế: Mất thông tin thứ tự từ
- Cải tiến: Bag of **n-grams (of tokens)**

- N-grams

- N-từ liên tiếp



Biểu diễn văn bản thành vector đặc trưng

□ Term Frequency (TF):

- Tần suất xuất hiện của các từ / ngữ (term) để biểu diễn

	t					
d	good	movie	not	a	did	like
good movie	1	1	0	0	0	0
not a good movie	1	1	1	1	0	0
did not like	0	0	1	0	1	1

□ Một số cách tính giá trị TF

weighting scheme	TF weight
binary	0, 1
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$1 + \log(f_{t,d})$

Biểu diễn văn bản thành vector đặc trưng

```

from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(tokenizer=word_tokenize)
train_data = [
    "good movie",
    "not a good movie movie",
    "did not like"
]
X_train_counts = count_vect.fit_transform(train_data)
print("====")
print(type(X_train_counts))
print(X_train_counts)

print("====")
print(type(X_train_counts.toarray()))
print(X_train_counts.toarray())
print("====")
print(count_vect.get_feature_names())

```

=====
 <class 'scipy.sparse.csr.csr_matrix'>
 (0, 2) 1
 (0, 4) 1
 (1, 2) 1
 (1, 4) 2
 (1, 5) 1
 (1, 0) 1
 (2, 5) 1
 (2, 1) 1
 (2, 3) 1
 =====
 <class 'numpy.ndarray'>
 [[0 0 1 0 1 0]
 [1 0 1 0 2 1]
 [0 1 0 1 0 1]]
 =====
 ['a', 'did', 'good', 'like', 'movie', 'not']

Chuyển một văn bản mới thành vector

```

print(count_vect.transform(["like a good"]).toarray())
[[1 0 1 1 0 0]]

```

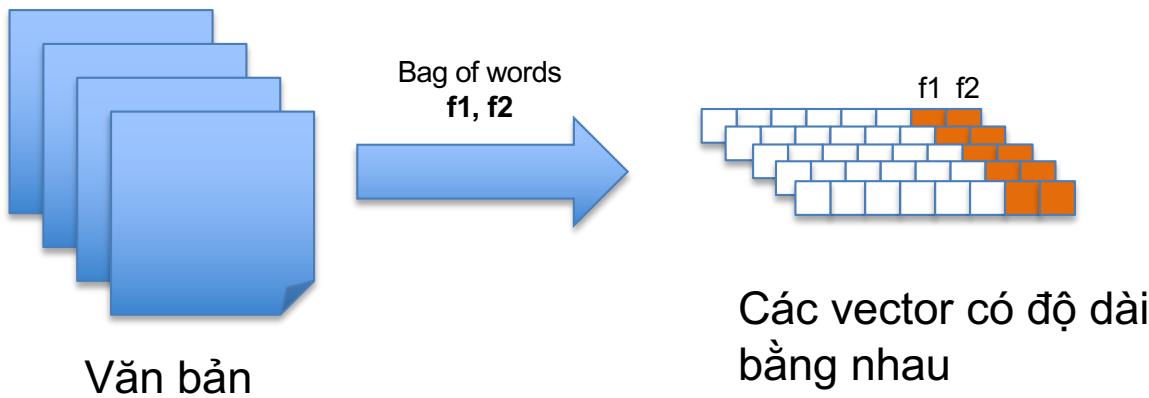
Biểu diễn văn bản thành vector đặc trưng

- CountVectorizer:
 - https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
- Bài tập:
 - Tìm hiểu thêm về các tham số của **CountVectorizer**
 - Tự cài đặt CountVectorizer không sử dụng sklearn
 - Tìm hiểu sử dụng, các tham số của: **TfidfTransformer**
 - https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
 - Tự cài đặt **TfidfTransformer**



Biểu diễn văn bản thành vector đặc trưng

- Bổ sung các đặc trưng khác:
 - Ngoài các đặc trưng về mặt từ vựng, có thể rút trích các đặc trưng khác
 - VD: đặc trưng cú pháp, ngữ nghĩa, chứa các ký tự / từ đặc biệt, độ dài, ...



Ví dụ:

```
def f1(text):
    icon = "😊"
    return icon in text
```

Biểu diễn nhãn và dữ liệu

- Nhãn : Labels / Categories

- Chuyển sang chỉ số:

- Phân loại 2 lớp → 0, 1
 - Phân loại 5 lớp → 0, 1, 2, 3, 4

Dữ liệu huấn luyện

```
("good movie", "POSITIVE"),
("not a good movie movie", "POSITIVE"),
("did not like", "NEGATIVE")
```

Kích thước dữ liệu huấn luyện: N = 3 mẫu

Kết quả biểu diễn bằng word count

Số dòng = N, Số cột = Số đặc trưng

```
-----
<class 'numpy.ndarray'>
[[0 1 0 1 0]
 [0 1 0 2 1]
 [1 0 1 0 1]]
=====
['did', 'good', 'like', 'movie', 'not']
```

Nhãn [1, 1, 0]

Đầu vào phổ biến cho các thuật toán phân loại văn bản



Thuật toán phân loại

Thuật toán:

- Naïve Bayes
- SGDClassifier
- Logistic Regression
- Decision Tree
- Random Forest
- SVM
- ...

Thư viện hỗ trợ: **sklearn**, ...

- from sklearn.naive_bayes import **MultinomialNB**
- from sklearn.linear_model import **SGDClassifier**
- from sklearn import svm:
 - svm.**LinearSVC()**
 - svm.**SVC**(kernel='linear')
 - svm.**SVC**(kernel='rbf')

```
pip install scikit-learn
```



Huấn luyện

- sklearn:
 - Thư viện các thuật toán học máy đã được chuẩn hoá về mặt giao tiếp → dễ sử dụng
- Huấn luyện:

```
classifier = XYZClassifier()  
classifier.fit(X_train, train_target)
```

- Sử dụng: classifier.**predict(X_new)** → class
- Đánh giá:
 - **Nguyên tắc chung**: hệ thống nào đáp ứng đúng mong đợi của người dùng hơn thì hệ thống đó tốt hơn.
 - Có nhiều độ đo thực hiện việc đánh giá hệ thống → Phải chọn độ đo phù hợp



Demo



Khai thác ngũ liệu văn bản và UD

Biểu diễn và phân loại văn bản (Phần 2)

Nguyễn Trường Sơn
ntson@fit.hcmus.edu.vn



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung

- Ví dụ mở đầu
- Các phương pháp cho bài toán phân loại văn bản
 - Sử dụng luật
 - Sử dụng các cách tiếp cận máy học truyền thống
 - Biểu diễn văn bản
 - Xây dựng mô hình huấn luyện
- Cách đánh giá một hệ thống phân loại văn bản
 - Accuracy
 - Precision / Recall / F1
- Các phương pháp tinh chỉnh tham số: Hyper parameter tuning
- Cross Validation
- Sử dụng các phương pháp học sâu

Tóm tắt phần trước

- Chuẩn bị dữ liệu
 - Dữ liệu và nhãn
- Chọn phương pháp để biểu diễn văn bản và nhãn:
 - CountVectorizer
 - TfidfTransformer
 - Sử dụng unigram, bigram, ...
- Chọn thuật toán để phân loại:
 - SGDClassifier
 - LinearSVC
 - SVC
 - LogisticRegression
- Huấn luyện
- Đánh giá kết quả mô hình

Cách thực nghiệm và chọn lựa mô hình

- Xây dựng mô hình có x (giả sử $x=2$) giai đoạn thực hiện:
 - Giai đoạn 1: Biểu diễn văn bản
 - Có **m** cấu hình cần thử
 - Giai đoạn 2: Huấn luyện mô hình
 - Có **n** cấu hình cần thử

==> Tổng tất **m** x **n** thực nghiệm cần chạy

Chọn lựa cấu hình/phương pháp cho kết quả tốt nhất



Đánh giá hệ thống phân loại văn bản

- Đánh giá:
 - **Nguyên tắc chung:** hệ thống nào đáp ứng đúng mong đợi của người dùng hơn thì hệ thống đó tốt hơn.
 - Có nhiều độ đo thực hiện việc đánh giá hệ thống → Phải chọn độ đo phù hợp == Độ đo phản ánh đúng mong đợi của người dùng
 - Vd: Không phải lúc nào độ chính xác cũng sử dụng được
- Độ đo phổ biến:
 - Độ chính xác (Accuracy)
 - Precision / Recall / F1



Đánh giá hệ thống phân loại văn bản

Chuẩn bị dữ liệu / Chọn mô hình biểu diễn văn bản / chọn lựa mô hình phân lớp

Huấn luyện

Dự đoán và đánh giá

```

from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

ngram_range = (1,1)
use_idf = False
# step 1
count_vect = CountVectorizer(ngram_range=ngram_range)
X_train_counts = count_vect.fit_transform(train_data)
transformer = TfidfTransformer(use_idf=use_idf).fit(X_train_counts)
X_train = transformer.transform(X_train_counts)
print(X_train.shape)
clf = svm.LinearSVC()
clf.fit(X_train, train_target)

# step3: evaluation

print("Gold/Ground Truth Label:")
print(test_target[:30], "...")
print([id2label[x] for x in test_target[:10]], "...")

X_new_counts = count_vect.transform(test_data)
X_new = transformer.transform(X_new_counts)
predicted = clf.predict(X_new)
print("\nNumber Item Predicted:", len(predicted))
print("System / Predicted Label:")
print(list(predicted[:30]), "...")
ncorrect = sum([y_pred == y for y_pred, y in zip(predicted, test_target)])
accuracy = ncorrect / len(test_target)

print("\nResult:")
print(" ==> accuracy", accuracy)

```

Accuarcy

Kết quả dự đoán cho mô hình phân lớp câu hỏi:

(5452, 8410)

Gold/Ground Truth Label:

[1, 5, 3, 0, 1, 1, 3, 2, 0, 0, 5, 3, 1, 3, 1, 1, 2, 3, 0, 1, 3, 0, 5, 0, 0, 3, 0, 5, 5, 5] ...
['NUM', 'LOC', 'HUM', 'DESC', 'NUM', 'NUM', 'HUM', 'ENTY', 'DESC', 'DESC'] ...

Number Item Predicted: 500

System / Predicted Label:

[1, 5, 3, 0, 1, 1, 3, 0, 0, 5, 0, 1, 3, 1, 1, 5, 3, 0, 1, 3, 0, 5, 0, 0, 3, 0, 0, 5, 0] ...

Result:

==> accuracy 0.87

Accuracy =

$$\frac{\text{Số mẫu dự đoán đúng}}{\text{Tổng số mẫu}}$$



Precision / Recall / F1

□ Xét trên từng lớp:

□ Cách tính 1:

- Xét một lớp **C** cần đánh giá
- Cho:
 - **Tập dữ liệu gồm N tài liệu**
 - **M**: là số tài liệu có nhãn **C** ($M \leq N$) do chuyên gia gán nhãn (ground truth)
 - **K**: là số tài liệu do hệ thống gán nhãn **C**
 - **L**: là số tài liệu được gán nhãn đúng (hệ thống và chuyên gia giống nhau)
 - Precision: Độ đo sự chính xác = tỉ lệ nhãn lớp **C** được hệ thống tìm thấy đúng so với số lượng nhãn mà hệ thống gán nhãn **C**

$$\text{Precision} = \frac{L}{K} \quad \text{Recall} = \frac{L}{M}$$

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

Precision / Recall / F1

- ☐ Xét trên từng lớp:

Ground Truth: ['hum', 'hum', 'loc', 'loc', 'hum', 'hum']
System predict: ['hum', 'hum', 'loc', 'loc', 'loc', 'hum']

	M=số tài liệu có nhãn X	K=số tài liệu mà hệ thống gán nhãn X	L=số tài liệu hệ thống gán nhãn đúng	Precision	Recall	F1
hum	4	3	3	3/3=1.0	3/4=0.75	0.86
loc	2	3	2	2/3=0.67	2/2=1.0	0.80



Precision / Recall / F1

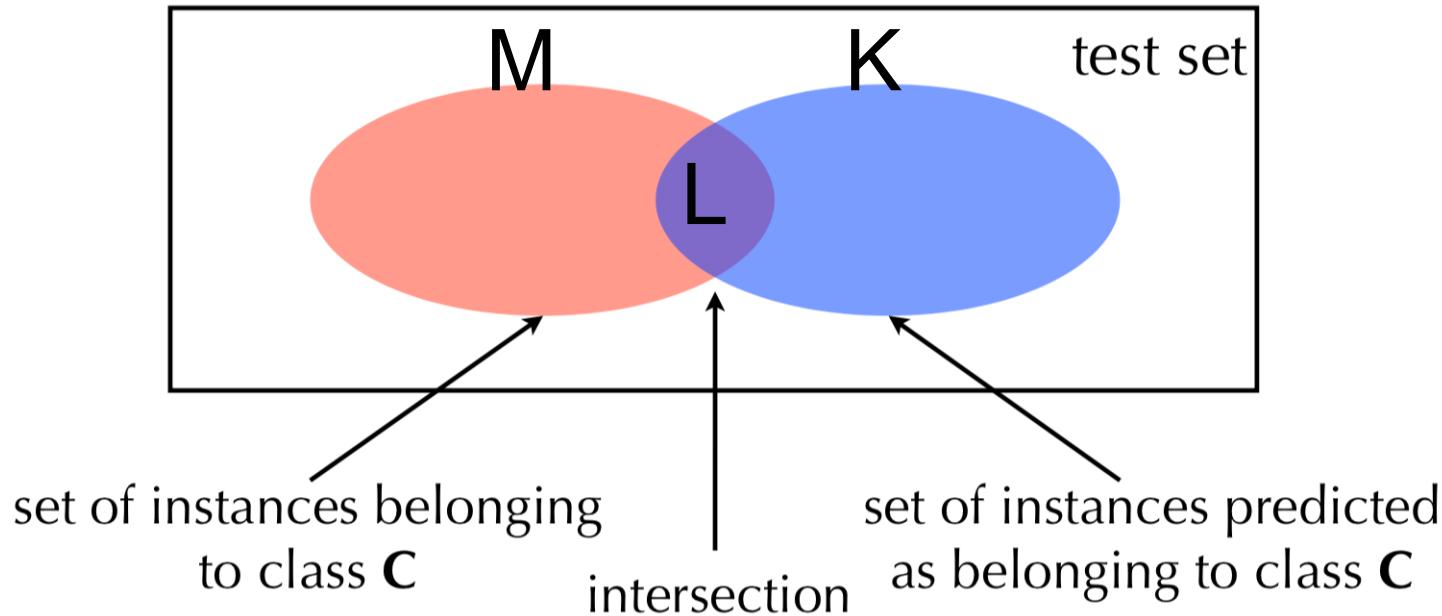
- Xét trên từng lớp:
 - Cách tính 2: Dựa trên confusion matrix

Ground Truth : ['hum', 'hum', 'loc', 'loc', 'hum', 'hum']
 System predict ['hum', 'hum', 'loc', 'loc', 'loc', 'hum']

		System		
		hum	loc	
Growth Truth	hum	3	1	
	loc	0	2	
				Tổng số nhãn hum chuyên gia gán nhãn=4
				Tổng số nhãn hum mà hệ thống dự đoán=3

Precision / Recall / F1

- ☐ Xét trên từng lớp C:



$$\text{Precision} = \frac{L}{K}$$

$$\text{Recall} = \frac{L}{M}$$

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

Precision / Recall / F1

```
[184]: from sklearn import metrics
labels=["hum", "loc"]
y_true=['hum', 'hum', 'loc', 'loc', 'hum', 'hum']
y_pred=['hum', 'hum', 'loc', 'loc', 'loc', 'hum']
print(metrics.classification_report(y_true, y_pred, target_names=labels))
cm = metrics.confusion_matrix(
    y_true=y_true,
    y_pred=y_pred,
    labels=labels
)
print(cm)
```

	precision	recall	f1-score	support
hum	1.00	0.75	0.86	4
loc	0.67	1.00	0.80	2
accuracy			0.83	6
macro avg	0.83	0.88	0.83	6
weighted avg	0.89	0.83	0.84	6

[[3 1]
[0 2]]

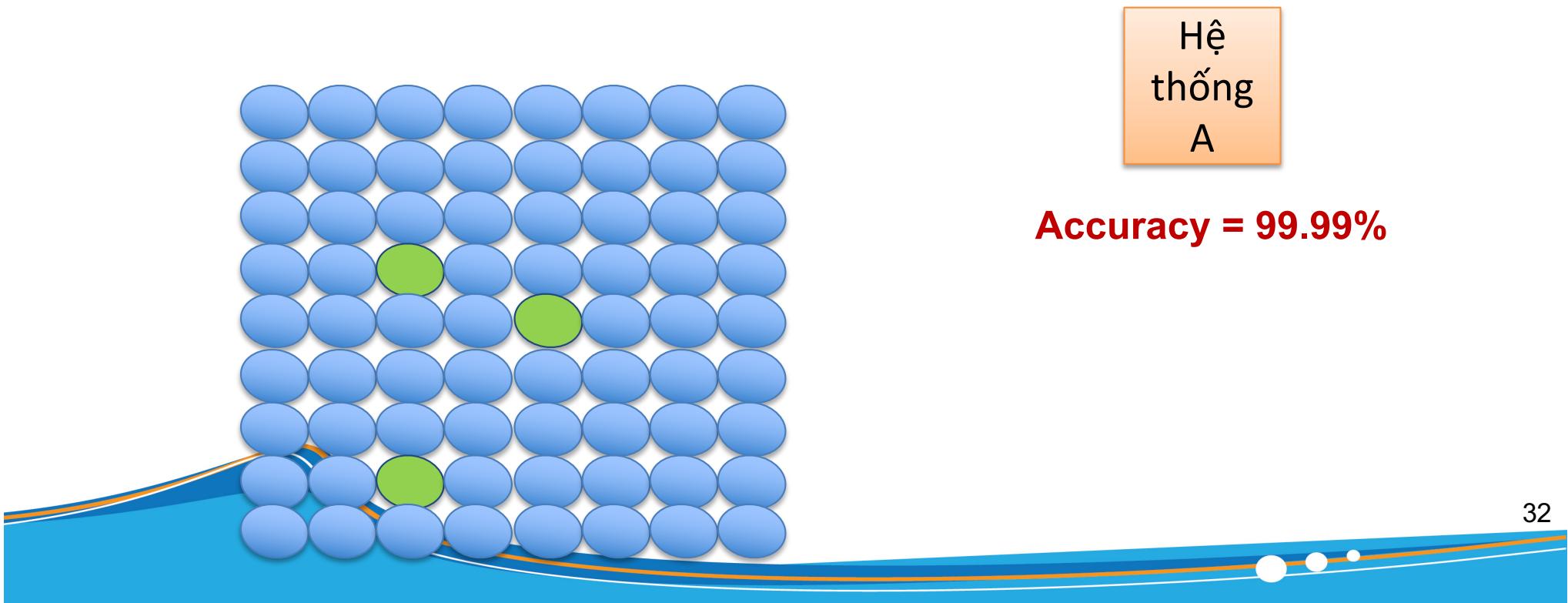
Growth Truth

System

	hum	loc
hum	3	1
loc	0	2

So sánh Accuracy, Precision, Recall, F1

- **Accuracy:** không đánh giá đúng trong những bài toán kiểu “to look for a needle in a haystack” = tìm kim đáy bể
 - Tập test có 1 triệu phần tử, có 50 phần tử cần tìm
 - Vd: 1 triệu hồ sơ bệnh án / tìm ra các bệnh nhân mang bệnh di truyền X (tỉ lệ thường là 0.005%)



So sánh Accuracy / Precision, Recall, F1

- What assumption(s) does accuracy make?
- It assumes that all prediction errors are equally bad
- Oftentimes, we care more about one class than the others
- If so, the class of interest is usually the minority class
- We are looking for the “needles in the haystack”
- In this case, accuracy is not a good evaluation metric
- There are metrics that provide more insight into per-class performance



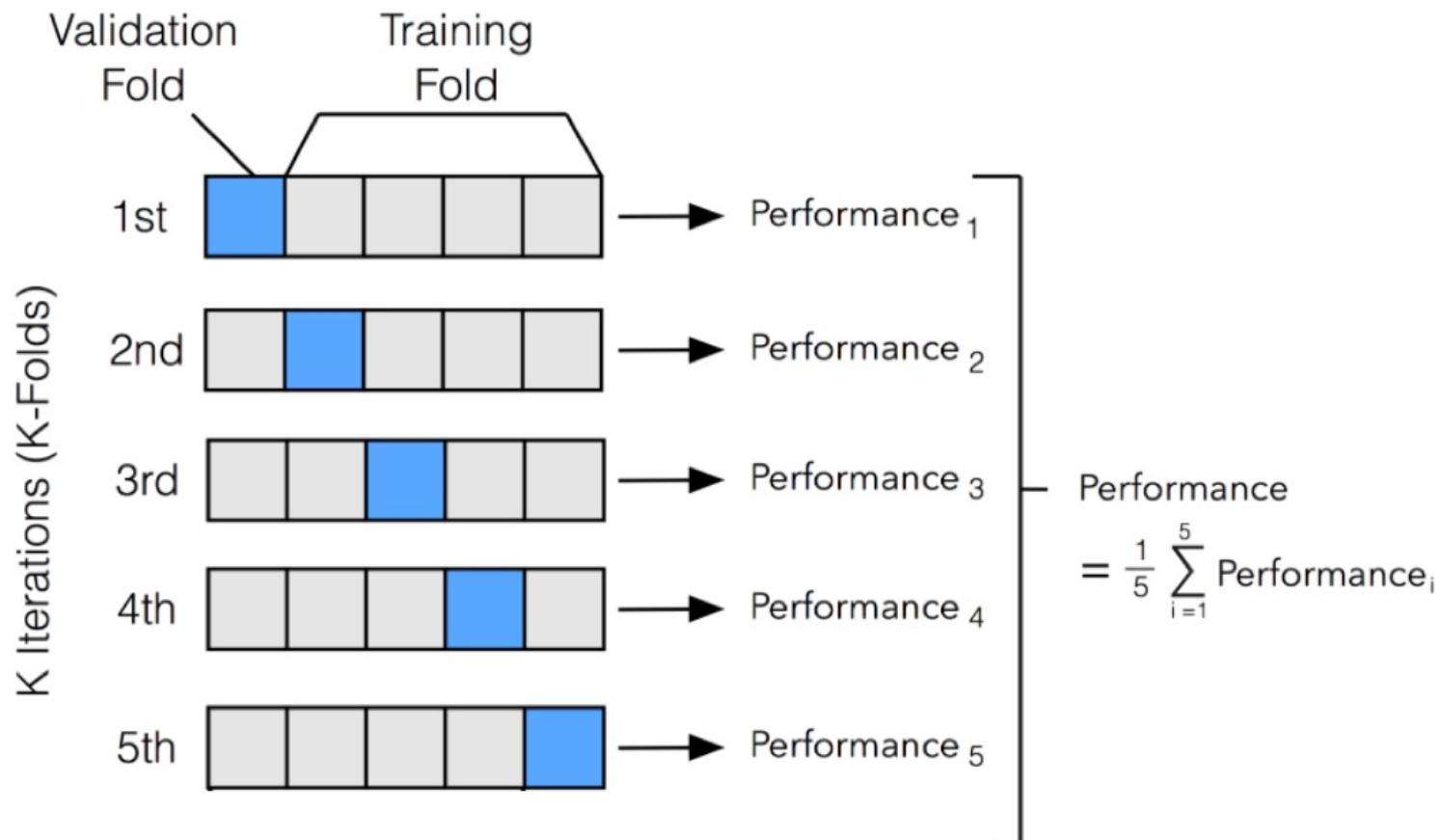
Cross validation

- Hiện tượng:
 - Hệ thống/phương pháp X cho kết quả F1=90% trên tập **test**
 - Hệ thống/phương pháp Y cho kết quả F1=85% trên tập **test**
- Có thể kết luận X tốt hơn Y ?
 - Có thể X tốt hơn Y trong bộ test đã cho, nhưng không đảm bảo trên bộ test khác
 - Trong trường hợp tập test chưa được biết trước → Làm sao để chọn lựa X hoặc Y ?
- cross-validation = giúp đánh giá chất lượng của một hệ thống / phương pháp



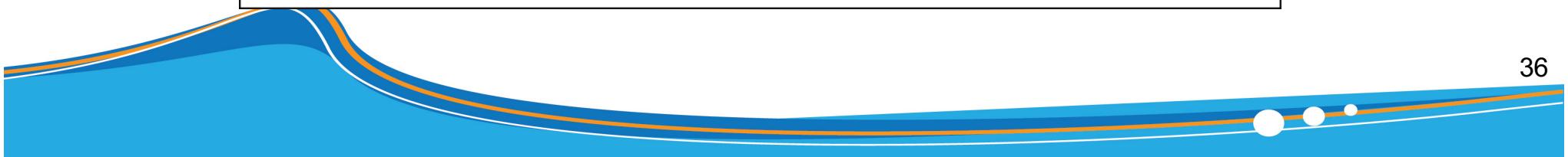
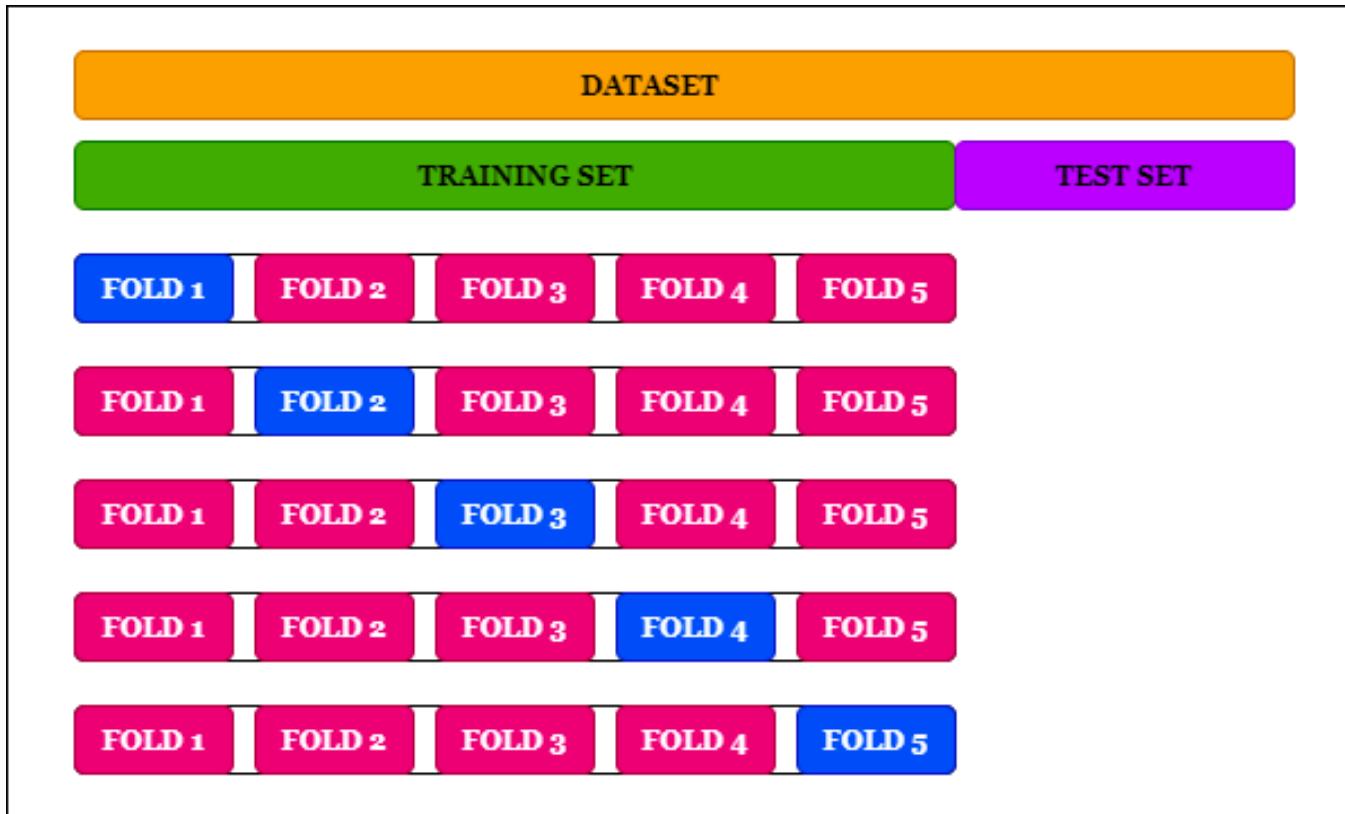
Cross validation

- Cách thực hiện k-fold cross validation:
 - Chia tập dữ liệu ra làm k phần:



Cross validation

- Cách thực hiện k-fold cross validation:
 - Chia tập dữ liệu ra làm k phần:



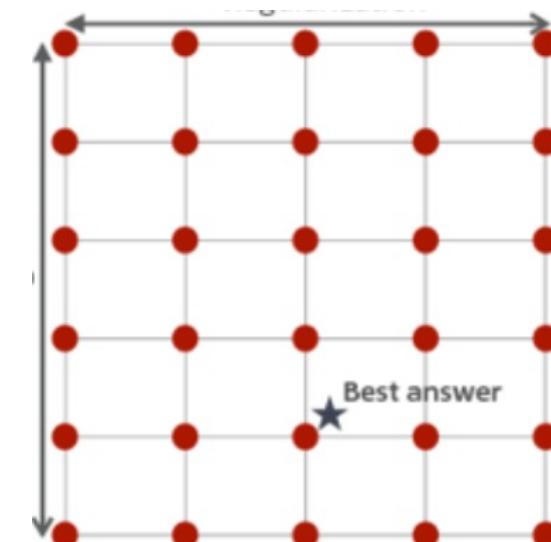
Train / Validate / Test set

- Train / Validate:
 - Sử dụng để huấn luyện và chọn tham số cho mô hình
- Nếu không có tập validate → tự tạo hoặc sử dụng -fold cross validation để chọn tham số tốt nhất của mô hình
- Test:
 - Không dùng vào trong suốt quá trình giải bài toán
 - Kết quả đánh giá cuối cùng trên tập này sẽ là khách quan nhất

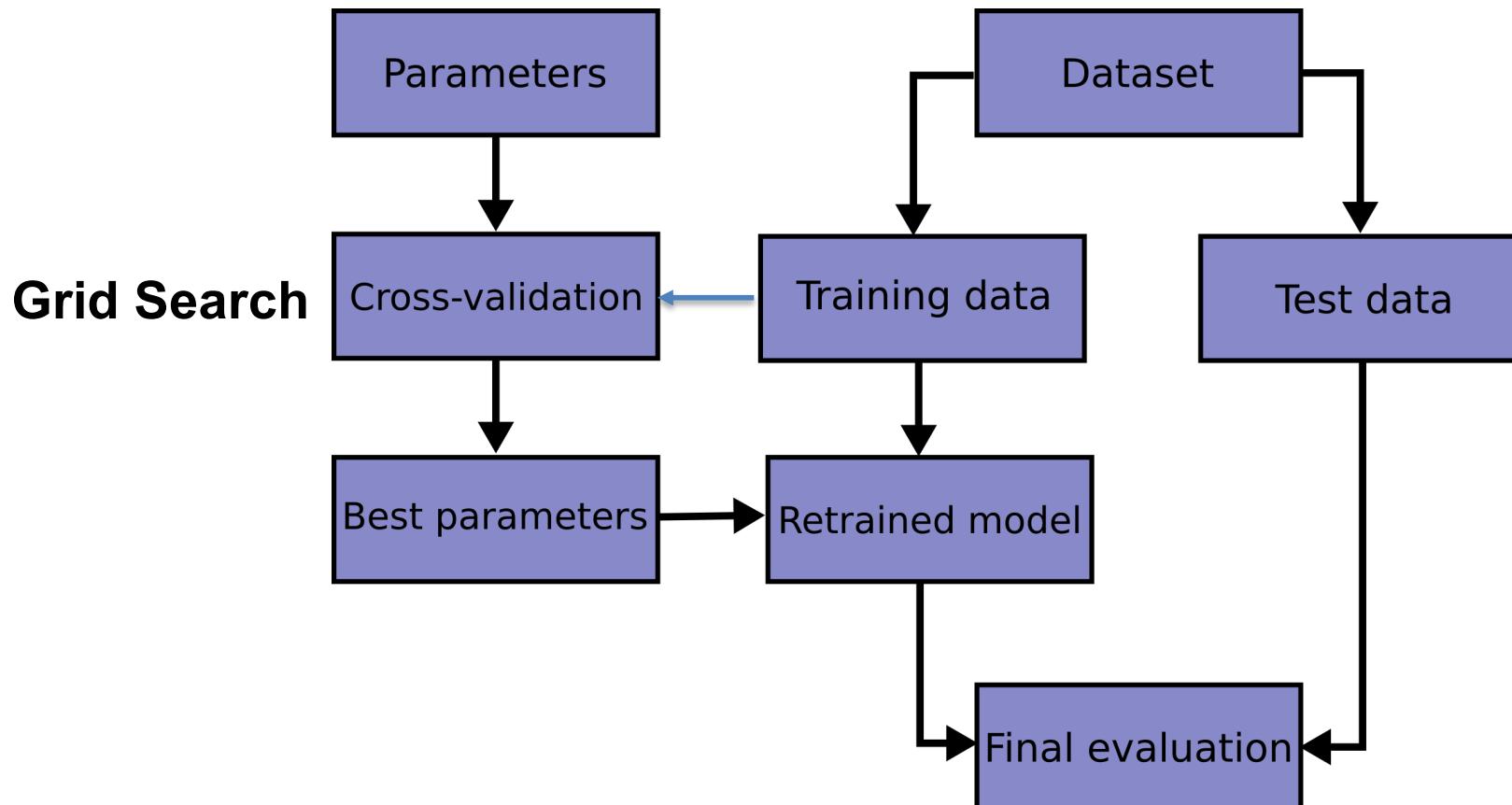


Tìm tham số tốt nhất của mô hình (Hyper parameters tuning)

- Giả sử một mô hình có 2 tham số cần chọn lựa:
 - Những tham số này cố định không điều chỉnh được trong quá trình huấn luyện
 - Số layer của mạng deep learning
 - Phương pháp biểu diễn văn bản --> vector:
 - count, tf, tfidf
 - Unigram, bigram, ...
 - a nằm trong khoảng [0, 1]
 - b nằm trong các giá trị [x, y, z, ...]
 - ...
- Phương pháp tìm a, b tốt nhất:
 - Phương pháp đơn giản: Grid Search
 - Phương pháp khác: GA, ...



Cross validation



https://scikit-learn.org/stable/modules/cross_validation.html

Grid Search

□ Before:

```

from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

C = 1.0
use_idf = False
ngram_range=(1, 1)

text_clf = Pipeline([
    ('vect', CountVectorizer(ngram_range=ngram_range)),
    ('tfidf', TfidfTransformer(use_idf=use_idf)),
    ('clf', svm.LinearSVC(C=C)),
])

```

```
text_clf.fit(train_data, train_target)
```

```

: predicted = text_clf.predict(test_data)
: ncorrect = sum([y_pred == y for y_pred, y in zip(predicted, test_target)])
: accuracy = ncorrect / len(test_target)
: accuracy

```

```
: 0.878
```

Grid Search

□ Doing Grid Search:

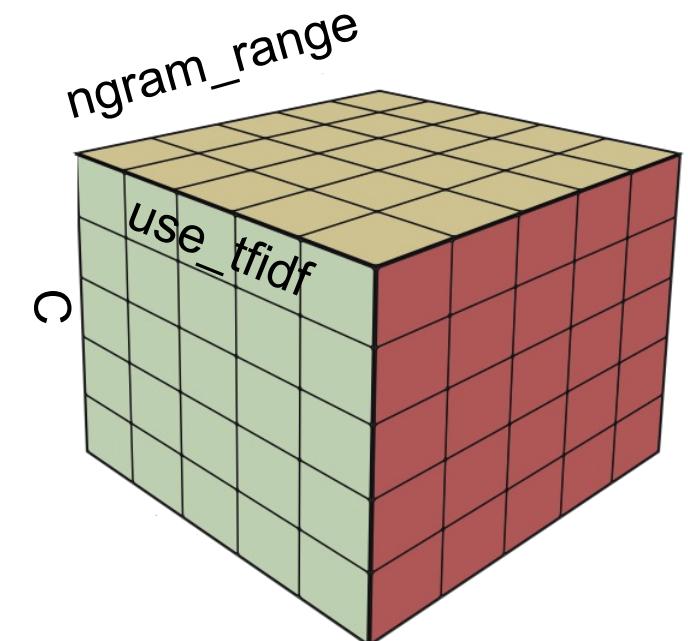
```
from sklearn.model_selection import GridSearchCV
parameters = {
    'vect_ngram_range': [(1, 1), (1, 2), (1, 3)],
    'tfidf_use_idf': (True, False),
    'clf_C': (1.0, 2.0, 3.0)
}
gs_clf = GridSearchCV(text_clf, parameters, cv=5, n_jobs=-1)
gs_clf = gs_clf.fit(train_data, train_target)
```

```
gs_clf.best_score_
```

```
0.8637176565561434
```

```
for param_name in sorted(parameters.keys()):
    print("%s: %r" % (param_name, gs_clf.best_params_[param_name]))
```

```
clf_C: 3.0
tfidf_use_idf: True
vect_ngram_range: (1, 2)
```



Grid Search

□ After

```
from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

C = 3.0
use_idf = True
ngram_range=(1, 2)
```

```
text_clf = Pipeline([
    ('vect', CountVectorizer(ngram_range=ngram_range)),
    ('tfidf', TfidfTransformer(use_idf=use_idf)),
    ('clf', svm.LinearSVC(C=C)),
])
```

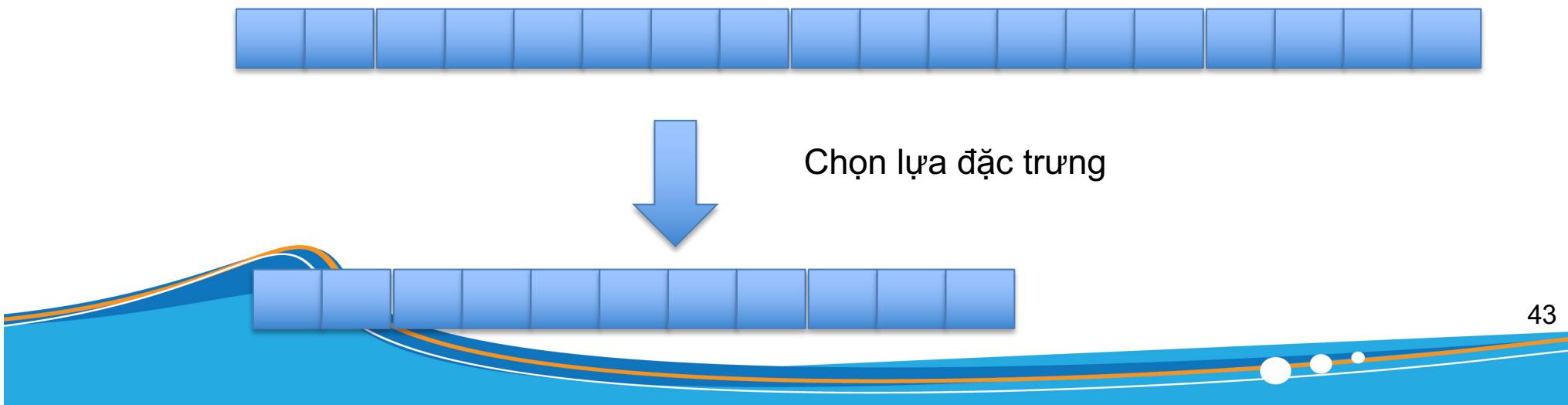
```
text_clf.fit(train_data, train_target)
```

```
predicted = text_clf.predict(test_data)
incorrect = sum([y_pred == y for y_pred, y in zip(predicted, test_target)])
accuracy = incorrect / len(test_target)
accuracy
```



Feature Selection

- Feature Selection: Chọn lựa đặc trưng (~Dimension Reduction)
 - Chọn lọc một tập con chứa các thuộc tính liên quan để sử dụng trong quá trình xây dựng mô hình.
 - Chọn các đặc trưng có giá trị, loại bỏ các đặc trưng gây nhiễu ảnh hưởng xấu tới việc xây dựng mô hình
 - Giảm kích thước tập đặc trưng → Huấn luyện nhanh hơn
Biểu diễn văn bản sử dụng bag of words (unigram, bigram, ...)



Feature Selection

- Feature selection methods can be classified into 4 categories. Filter, Wrapper, Embedded, and Hybrid methods.
 - **Filter** perform a statistical analysis over the feature space to select a discriminative subset of features.
 - **Wrapper** approach choose various subset of features are first identified then evaluated using classifiers.
 - The **embedded** approach the feature selection process is embedded into training phase of the classification.
 - **Hybrid** approach takes advantages of both filter and wrapper approaches.



Feature Selection

□ Filter → nhanh

Compute chi-squared stats between each non-negative feature and class.

This score can be used to select the `n_features` features with the highest values for the test chi-squared statistic from `X`, which must contain only non-negative features such as booleans or frequencies (e.g., term counts in document classification), relative to the classes.

Recall that the chi-square test measures dependence between stochastic variables, so using this function “weeds out” the features that are the most likely to be independent of class and therefore irrelevant for classification.

□ Wrapper → Châm

Feature ranking with recursive feature elimination.

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute or callable. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Feature Selection

```
from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.feature_selection import SelectKBest, chi2
C = 3.0
use_idf = True
ngram_range=(1, 2)

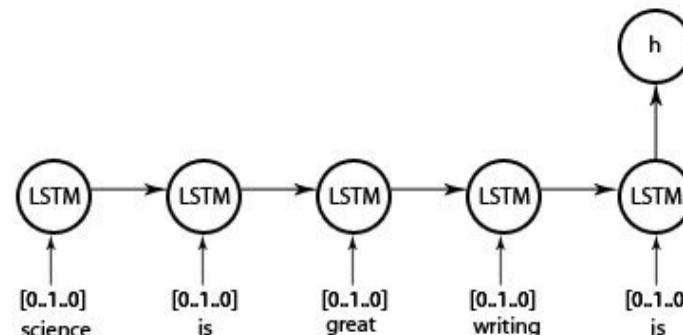
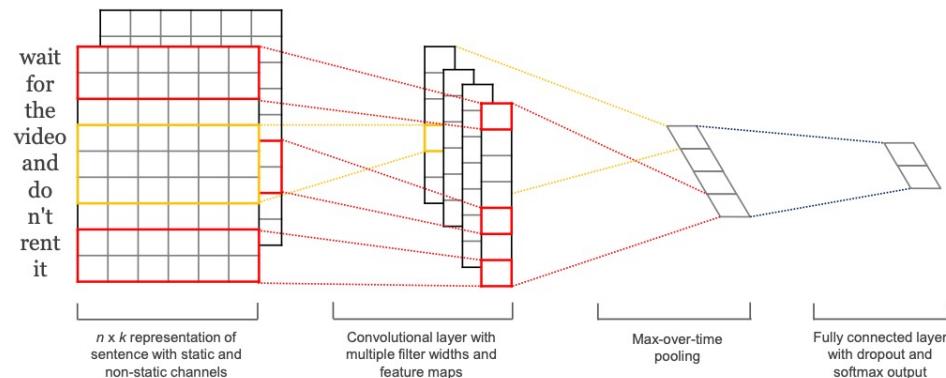
text_clf = Pipeline([
    ('vect', CountVectorizer(ngram_range=ngram_range)),
    ('tfidf', TfidfTransformer(use_idf=use_idf)),
    ('feature_selection', SelectKBest(chi2, k=12000)),
    ('clf', svm.LinearSVC(C=C)),
])
```

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html



Các mô hình phân lớp văn bản khác

- Convolutional Neural Networks for Sentence Classification (Yoon Kim)



- Text classification with LSTM:
 - https://www.tensorflow.org/text/tutorials/text_classification_rnn
- Text classification with Bert:
 - https://www.tensorflow.org/text/tutorials/classify_text_with_bert
- Text classification with Hugging face:
 - https://huggingface.co/docs/transformers/tasks/sequence_classification

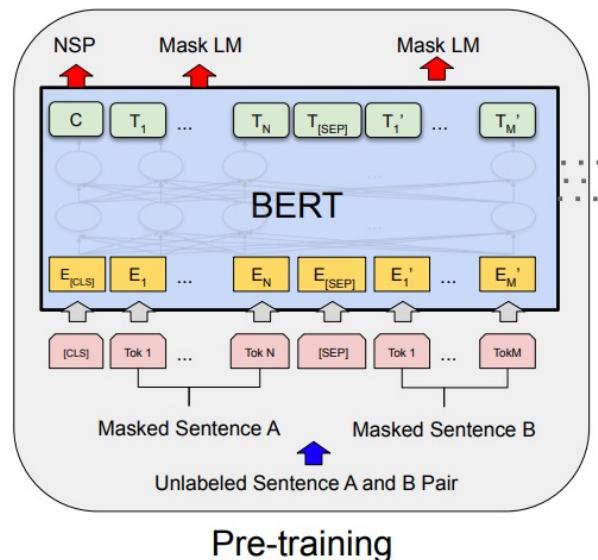
Các mô hình phân lớp văn bản khác

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

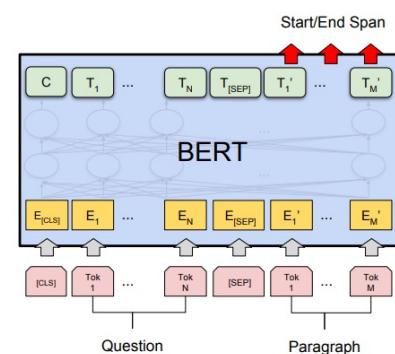
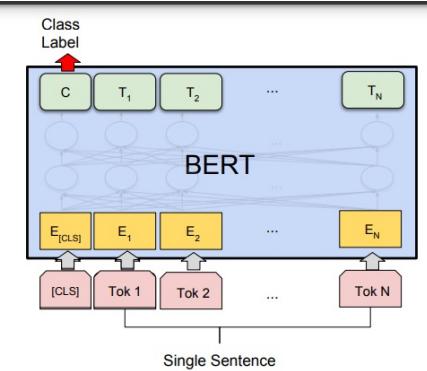
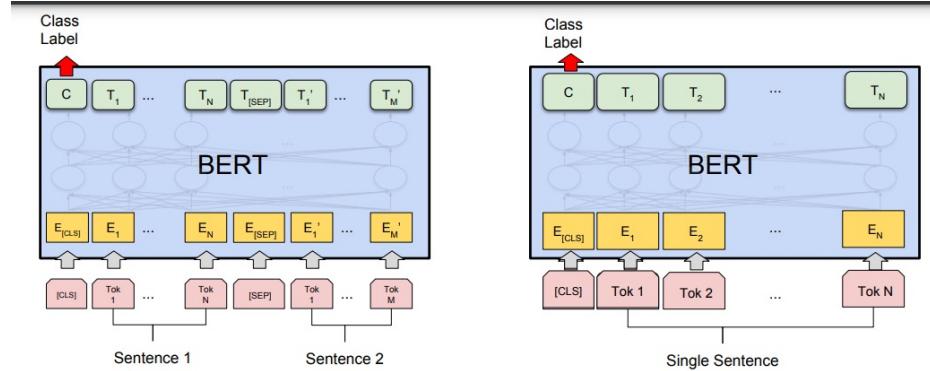
Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

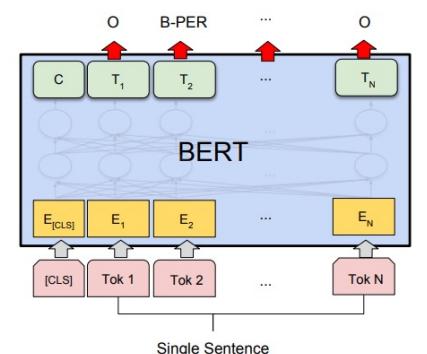
{jacobdevlin, mingweichang, kentonl, kristout}@google.com



<https://github.com/google-research/bert>



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Một biến thể của phân loại văn bản

- Phân loại cặp văn bản:
 - Đầu vào: cặp văn bản (t_1, t_2)
 - Đầu ra: Nhãn phân loại

- Một số bài toán tiêu biểu:
 - Nhận diện suy diễn văn bản
 - Xác định tương đồng / đồng nghĩa văn bản
 - Q-A relationship



Tóm tắt

- Cách đánh giá một hệ thống phân loại văn bản
 - Accuracy
 - Precision / Recall / F1
- Hyper parameter tuning
 - Grid Search
- Cross Validation
- Feature Selection



Bài tập

- Phân loại câu hỏi tiếng Việt:

- Bài toán:

- Đầu vào: **Câu hỏi**
 - Đầu ra: **Loại của câu hỏi**



Một số bước xử lý khác

□ Tách từ: Cho tiếng Anh

```
from nltk.tokenize import word_tokenize
text = "After sleeping for four hours, he decided to sleep for
tokens = word_tokenize(text)
print(tokens)

['After', 'sleeping', 'for', 'four', 'hours', ',', 'he', 'deci
ded', 'to', 'sleep', 'for', 'another', 'four']
```

[Google: NLTK Tokenize, ...](#)

□ Loại bỏ những từ không cần thiết:

- Từ không cần thiết == Stop words: Từ không có ý nghĩa cho bài toán đang xét

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent = "This is a sample sentence"
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(example_sent)
filtered_sentence = [w for w in word_tokens if not w in stop_words]
print(word_tokens)
print(filtered_sentence)

['This', 'is', 'a', 'sample', 'sentence']
['This', 'sample', 'sentence']
```

list comprehension in python

```
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
```

Bài tập

- Tìm 1 dataset về sentiment analysis Tiếng Việt, chia thành các tập huấn luyện, phát triển và kiểm thử
- Thủ nghiệm xây dựng các mô hình phân loại
- Thực nghiệm, tinh chỉnh tham số và đánh giá các mô hình.
- Một số kết luận

