Xử lý văn bản cơ bản

Biểu thức chính quy
(Regular Expression)

Biểu thức chính quy (regular expression)

- Một loại ngôn ngữ hình thức để biểu diễn các chuỗi văn bản
- Làm thế nào để tìm ra những chuỗi sau?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

Regular Expression: Disjunction

- Ký tự trong ngoặc vuông []

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Khoảng giá trị `[A-Z]`

Pattern	Matches	
`[A-Z]`	An upper case letter	`Drenched Blossoms`
`[a-z]`	A lower case letter	`my beans were impatient`
`[0-9]`	A single digit	`Chapter 1: Down the Rabbit Hole`

Regular Expressions: Phủ định trong Disjunction

- Phủ định `[^Ss]`
  - Dấu mũ (carat) đầu tiên trong [] là dấu phủ định, các dấu mũ bên trong không phải, mà là ký tự dấu mũ.

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither 'S' nor 's'	`I have no exquisite reason"`
`[^e^]`	Neither e nor ^	`Look here`
`a^b`	The pattern a carat b	`Look up a^b now`

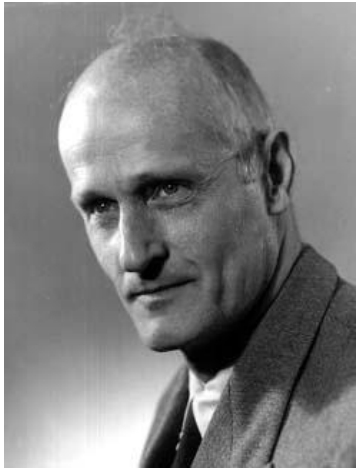
Regular Expressions: Nói thêm về Disjunction

- Woodchuck là tên khác của groundhog!
- Dùng dấu | cho disjunction

Pattern	Matches
`groundhog|woodchuck`	
`yours|mine`	yours mine
`a|b|c`	= [abc]
`[gG]roundhog|[Ww]oodchuck`	

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	KT liền trước là không bắt buộc (optional)	color colour
oo*h!	0 hoặc nhiều KT liền trước	oh! ooh! oooh! ooooh!
o+h!	1 hoặc nhiều KT liền trước	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

Ví dụ

- Tìm thất cả các chữ “the” trong một văn bản.

```
the
```

Misses capitalized examples

```
[tT]he
```

Incorrectly returns `other` or `theology`

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

Lỗi so khớp

- Quá trình chúng ta làm trong slide trước dựa trên mục tiêu sửa dần 2 loại lỗi:
  - Khớp (match) những chuỗi mà đáng lẽ không nên match (there, then, other)
    - False positives (Type I)
  - Không khớp được những chuỗi mà đáng lẽ phải match (The)
    - False negatives (Type II)

Lỗi so khớp (tt)

- Trong XLNNTN chúng ta luôn phải giải quyết những loại lỗi này.
- Giảm tỉ lệ lỗi trong các ứng dụng thường dẫn đến 2 nỗ lực trái ngược nhau:
  - Tăng accuracy hay precision (giảm thiểu false positives)
  - Tăng độ phủ coverage hay recall (giảm thiểu false negatives).

Bài tập

Câu 1: Viết biểu thức chính qui để so khớp với những từ: woodchuck, woodchucks, Woodchuck và Woodchucks.

Câu 2: Viết biểu thức chính quy để so khớp với các số điện thoại Việt Nam.

Câu 3: Suy nghĩ ra 3 ứng dụng có thể giải quyết bằng biểu thức chính qui?

Sử dụng regular expression (online): https://rubular.com/

Tóm tắt

- Regular expression đóng vai trò rất quan trọng (hơn tưởng tượng)
  - Thường là mô hình đầu tiên cho bất kỳ tác vụ xử lý văn bản nào
- Đối với nhiều tác vụ khó, chúng ta sử dụng những bộ phân lớp dựa trên học máy (machine learning classifiers)
  - Nhưng RE vẫn được sử dụng làm đặc trưng (**features**) của những bộ phân lớp (classifiers).

Xử lý văn bản cơ bản

Tách từ

(Word tokenization)

Chuẩn hóa văn bản

- Mọi tác vụ XLNNTN đều cần bước chuẩn hóa:
  1. Tách từ
  2. Chuẩn hóa các dạng thức của từ
  3. Tách câu

Có bao nhiêu từ

- I do uh main- mainly business data processing
  - Phân mảnh (fragments), điểm dừng (filled pauses)
- Seuss's cat in the hat is different from other cats!
  - **Gốc từ (Lemma)**: Cùng gốc từ (stem), từ loại (part of speech), ngữ nghĩa tổng quát (word sense)
    - cat và cats = cùng gốc từ (lemma)
  - **Dạng từ (Wordform)**: là hình thức đầy đủ của từ (bao gồm những biến cách)
    - cat và cats = khác dạng từ (wordform)

Có bao nhiêu từ?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: một thành phần của từ vựng.
- **Token**: một thực thể (instance) của loại từ đó trong văn bản.
- Có bao nhiêu?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

Có bao nhiêu từ?

N = Số lượng tokens

V = từ vựng = tập hợp các loại

$|V|$ là kích thước của từ vựng

Corpus	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 triệu	20 ngàn
Shakespeare	884,000	31 ngàn
Google N-grams	1 ngàn tỉ	13 triệu

Vấn đề trong Tokenization

- Finland’s capital → Finland Finlands Finland’s ?
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → 1 hay 2 token?
- m.p.h., PhD. → ??

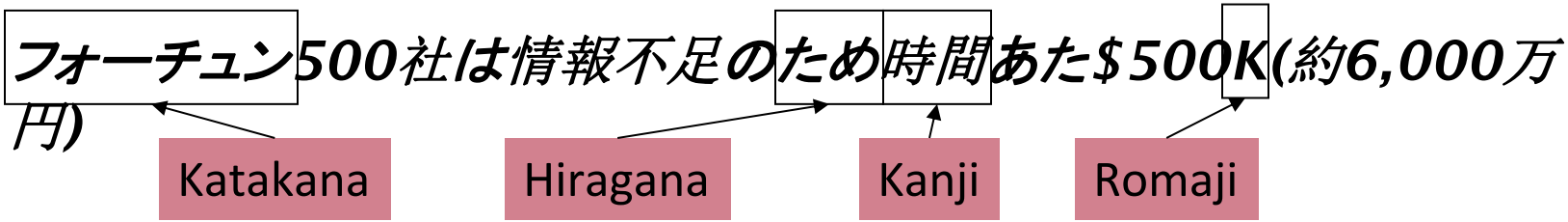
Tokenization: những thách thức riêng của từng ngôn ngữ

- Tiếng Pháp
  - ***L'ensemble*** → 1 hay 2 token?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Muốn xem 2 cụm này là một: ***l'ensemble*** và ***un ensemble***

- Tiếng Đức: Danh từ phức (noun compound) không được tách ra
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'nhân viên công ty bảo hiểm nhân thọ'
  - IR tiếng Đức cần bộ tách danh từ phức

Tokenization: những thách thức riêng của từng ngôn ngữ

- Tiếng Hoa, tiếng Nhật không có khoảng trắng giữa các từ
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Tiếng Nhật sử dụng nhiều bộ chữ trộn lẫn
  - Ngày tháng/ số lượng có format khác nhau



Người sử dụng có thể nhập query gồm toàn hiragana!

Tách từ trong tiếng Hoa

- Còn gọi là **Word Segmentation**
- Từ trong tiếng Hoa được tạo thành từ các ký tự (character)
  - Ký tự nói chung có 1 âm tiết (syllable) và 1 hình vị (morpheme).
  - Trung bình mỗi từ có 2.4 ký tự.
- Thuật toán tách từ baseline chuẩn:
  - Maximum Matching (còn gọi là thuật toán Greedy)

Thuật toán tách từ Maximum Matching

- Cho trước một chuỗi ký tự tiếng Hoa.

1) Khởi tạo một con trỏ (pointer) tại điểm đầu của chuỗi
2) Tìm từ dài nhất trong tự điển khớp với chuỗi bắt đầu từ vị trí con trỏ
3) Dời con trỏ sang phải qua từ đã tìm được
4) Quay lại bước 2

Minh họa thuật toán Max-match

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there / theta bled own there
- Không hoạt động tốt cho tiếng Anh!

- Nhưng hoạt động tốt đáng kinh ngạc cho tiếng Hoa.
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Thuật toán tách từ dựa trên xác suất hiện đại hoạt động tốt hơn.

Tiếng Việt

- Là ngôn ngữ đơn lập (Trung Quốc, Nhật, Thái, và Việt).
- Với các ngôn ngữ đơn lập, một từ có thể có một hoặc nhiều âm tiết.
- Vấn đề của bài toán tách từ là khử được sự nhập nhằng trong ranh giới từ.

Tiếng Việt – Từ vựng

- Tiếng Việt là ngôn ngữ không biến hình.
- Từ điển từ tiếng Việt (Vietlex): >40.000 từ, trong đó:
  - 81.55% âm tiết là từ : từ đơn
  - 15.69% các từ trong từ điển là từ đơn
  - 70.72% từ ghép có 2 âm tiết
  - 13.59% từ ghép ≥ 3 âm tiết
  - 1.04% từ ghép ≥ 4 âm tiết

Tiếng Việt – Từ vựng

Bảng 1. Độ dài của từ tính theo âm tiết

Độ dài	Số lượng	Tỉ lệ
1	6,303	15.69
2	28,416	70.72
3	2,259	5.62
4	2,784	6.93
5	419	1.04
Tổng	40,181	100

Tiếng Việt – Qui tắc cấu tạo từ

- Từ đơn: dùng một âm tiết làm một từ.
  - Ví dụ: tôi, bác, người, cây, hoa, đi, chạy, vì, đã, à, nhỉ, nhé...
- Từ ghép: tổ hợp (ghép) các âm tiết lại, giữa các âm tiết đó có quan hệ về nghĩa với nhau.
  - **Từ ghép đẳng lập**. các thành tố cấu tạo có quan hệ bình đẳng với nhau về nghĩa.
    - Ví dụ: chợ búa, bếp núc
  - **Từ ghép chính phụ**. các thành tố cấu tạo này phụ thuộc vào thành tố cấu tạo kia. Thành tố phụ có vai trò phân loại, chuyên biệt hoá và sắc thái hoá cho thành tố chính.
    - Ví dụ: tàu hoả, đường sắt, xấu bụng, tốt mã, ngay đơ, sưng vù...

Tiếng Việt – Qui tắc cấu tạo từ

- Từ láy: các yếu tố cấu tạo có thành phần ngữ âm được lặp lại; nhưng vừa lặp vừa biến đổi. Một từ được lặp lại cũng cho ta từ láy…
- Biến thể của từ: được coi là dạng lâm thời biến động hoặc dạng "lời nói" của từ.
  - Rút gọn một từ dài thành từ ngắn hơn
    - ki-lô-gam → ki lô/ kí lô
- Phá vỡ cấu trúc của từ, phân bố lại yếu tố tạo từ với những yếu tố khác ngoài từ chen vào. Ví dụ:
  - khổ sở → lo khổ lo sở
  - ngặt nghẽo → cười ngặt cười nghẽo
  - danh lợi + ham chuộng → ham danh chuộng lợi

Tiếng Việt – Qui tắc cấu tạo từ

- Các diễn tả gồm nhiều từ (vd, “bởi vì”) cũng được coi là 1 từ
- Tên riêng: tên người và vị trí được coi là 1 đơn vị từ vựng
- Các mẫu thường xuyên: số, thời gian

Xử lý văn bản cơ bản

Chuẩn hóa từ và đưa về gốc từ

Chuẩn hóa

- Sự cần thiết phải "chuẩn hóa (normalize)" từ khóa (term)
  - Truy vấn thông tin (Information Retrieval): văn bản được đánh chỉ mục (indexed text) và từ khóa truy vấn (query term) phải có cùng dạng thức.
    - Chúng ta muốn so khớp ***U.S.A.*** và ***USA***
- Chúng ta ngầm định nghĩa ra các lớp từ khóa tương đương nhau.
  - VD: xóa dấu chấm "." trong từ khóa
- Mở rộng từ khóa:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***

Chữ hoa, thường

- Ứng dụng như IR: chuyển tất cả các chữ thành chữ thường
  - Vì người sử dụng thường gõ chữ thường
  - Ngoại lệ: Chữ hoa ở giữa câu?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- Đối với bài toán Phân tích ý kiến (sentiment analysis), dịch máy (machine translation), rút trích thông tin (information extraction)
  - Chữ hoa, thường cũng hữu ích (***US*** và ***us***)

Đưa từ về nguyên mẫu (lemmatization)

- Chuyển dạng dẫn xuất của từ (inflection) hoặc các dạng khác trở về dạng nguyên mẫu.
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
- *the boy's cars are different colors → the boy car be different color*
- Đưa từ về nguyên mẫu (Lemmatization): phải tìm được dạng đúng của từ trong từ điển.
- Dịch máy
  - Spanish quiero (‘I want’), quieres (‘you want’) ❓ nguyên mẫu querer ‘want’

Hình thái từ (Morphology)

- **Hình vị (Morpheme)**:
  - Đơn vị có nghĩa nhỏ nhất tạo thành từ.
  - **Gốc từ (Stem)**: Đơn vị mang nghĩa cốt lõi
  - **Phụ tố (Affix)**: Những thứ được gắn thêm vào trước hoặc sau gốc từ
    - Thường đi kèm với chức năng ngữ pháp nhất định

Cắt từ để đưa về gốc từ (Stemming)

- Chuyển từ khóa về gốc từ trong truy vấn thông tin
- *Stemming* dùng phương pháp "thô" để cắt bỏ phụ tố
  - Phụ thuộc ngôn ngữ
  - e.g., ***automate(s), automatic, automation*** đều được chuyển thành ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Thuật toán Porter (phổ biến nhất cho tiếng Anh)

Step 1a

sses	→ ss	caresses	→ caress
ies	→ i	ponies	→ poni
ss	→ ss	caress	→ caress
s	→ ø	cats	→ cat

Step 1b

(*v*)ing	→ ø	walking	→ walk
		sing	→ sing
(*v*)ed	→ ø	plastered	→ plaster
…			

Step 2 (for long stems)

ational	→ ate	relational	→ relate
izer	→ ize	digitizer	→ digitize
ator	→ ate	operator	→ operate
…			

Step 3 (for longer stems)

al	→ ø	revival	→ reviv
able	→ ø	adjustable	→ adjust
ate	→ ø	activate	→ activ
…			

Xem hình thái từ trong một ngữ liệu
Tại sao chỉ bỏ -ing khi có 1 nguyên âm?

(*v*)ing → ø walking → walk

sing → sing

Một số ngôn ngữ cần xử lý hình thái từ phức tạp

- VD:
  - Tiếng Thổ Nhĩ Kỳ (Turkish)
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(behaving) as if you are among those whom we could not civilize'
  - Uygar `civilized' + las `become'
    + tir `cause' + ama `not able'
    + dik `past' + lar 'plural'
    + imiz 'p1pl' + dan 'abl'
    + mis 'past' + siniz '2pl' + casina 'as if'

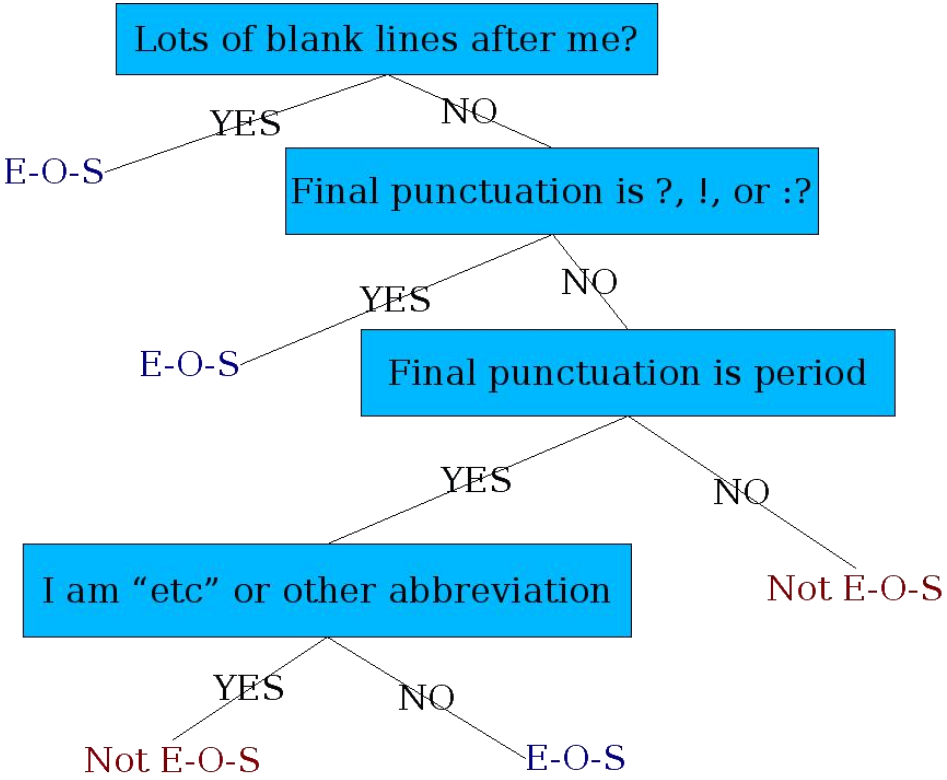
Xử lý văn bản cơ bản

Tách câu và cây quyết định

Tách câu

- !, ? tương đối rõ ràng.
- Dấu “.” cũng rất nhập nhằng
  - Ranh giới câu
  - Từ viết tắt (Abbreviations) như Inc. hay Dr.
  - Số như .02% hay 4.3
- Xây dựng một bộ phân lớp nhị phân
  - Xem xét một dấu “.”
  - Xác định xem nó là dấu chấm châu hay không: EndOfSentence/NotEndOfSentence
  - Bộ phân lớp: luật viết tay, regular expression, hay học máy

Xác định một từ có phải kết thúc câu bằng cây quyết định



Những đặc trưng phức tạp hơn

- Từ trước dấu chấm là hoa hay thường: Upper, Lower, Cap, Number
- Từ sau dấu chấm là hoa thay thường: Upper, Lower, Cap, Number

- Đặc trưng số
  - Chiều dài của từ trước dấu “.”
  - Xác suất(từ trước “.” xuất hiện ở cuối câu)
  - Xác suất(word after “.” xuất hiện ở đầu câu)

Cài đặt cây quyết định

- Cây quyết định chỉ là một lệnh if-then-else
- Thú vị: nghiên cứu cách chọn đặc trưng
- Dựng cấu trúc cây quyết định rất khó làm bằng tay
  - Cây quyết định xây dựng bằng tay chỉ khả thi khi đặc trưng rất đơn giản hoặc miền ứng dụng (domain) đơn giản
    - Đối với những đặc trưng số, rất khó để chọn ngưỡng
  - Thay vì vậy, cấu trúc cây quyết định có thể học tự động bằng phương pháp học máy dựa trên ngữ liệu huấn luyện (training corpus)

Cây quyết định và những bộ phân lớp khác

- Chúng ta có thể coi những câu hỏi là một cây quyết định
- Đặc trưng sử dụng bởi cây quyết định có thể sử dụng cho những loại bộ phân lớp khác.
  - Logistic regression
  - SVM
  - Neural Nets
  - etc.