

# Information Extraction

Lecture 2 – IE Scenario, Text Selection/Processing,  
Extraction of Closed & Regular Sets

CIS, LMU München

Winter Semester 2020-2021

Prof. Dr. Alexander Fraser, CIS

# Administravia I

- **Please check LSF** to make sure you are registered
  - Note that CIS students need to be registered for BOTH the Vorlesung and the Seminar (two registrations!)
- Later in the semester you will have to register yourself in LSF for the Klausur (and to get a grade in the Seminar)
  - Two "Klausur" registrations if you need both grades (most CISlers)

# Administravia II

- Seminars this week: Referat topics
  - Signup Monday at 18:00 by email \*to me\*, see my slides
- No seminars next week. Lecture will take place.

# Reading for next time

- Please read Sarawagi Chapter 2 for next time (rule-based NER)

# Outline

- IE Scenario
- Information Retrieval vs. Information Extraction
- Source selection
- Tokenization and normalization
- Extraction of entities in closed and regular sets
  - e.g., dates, country names

# Relation Extraction: Disease Outbreaks

May 19 1995, Atlanta -- The Centers for Disease Control and Prevention, which is in the front line of the world's response to the deadly Ebola epidemic in Zaire , is finding itself hard pressed to cope with the crisis...

**Information  
Extraction System**

<i>Date</i>	<i>Disease Name</i>	<i>Location</i>
Jan. 1995	Malaria	Ethiopia
July 1995	Mad Cow Disease	U.K.
Feb. 1995	Pneumonia	U.S.

# IE tasks

- Many IE tasks are defined like this:
  - Get me a database like this
  - For instance, let's say I want a database listing severe disease outbreaks by country and month/year
- Then you find a corpus containing this information
  - And run information extraction on it

# IE Scenarios

- Traditional Information Extraction
  - This will be the main focus in the course
  - Which templates we want is predefined
    - For our example: **disease outbreaks**
  - Instance types are predefined
    - For our example: **diseases, locations, dates**
  - Relation types are predefined
    - For our example, outbreak: **when, what, where?**
  - Corpus is often clearly specified
    - For our example: a **newspaper corpus** (e.g., the New York Times), with new articles appearing each day
- However, there are other interesting scenarios...
- Information Retrieval
  - Given an information need, find me documents that meet this need from a collection of documents
    - For instance: Google uses short queries representing an abstract information need to search the web
- Non-traditional IE
  - Two other interesting IE scenarios
    - Question answering
    - Structured summarization
  - Open IE
    - IE without predefined templates!
    - Will cover this at the end of the semester



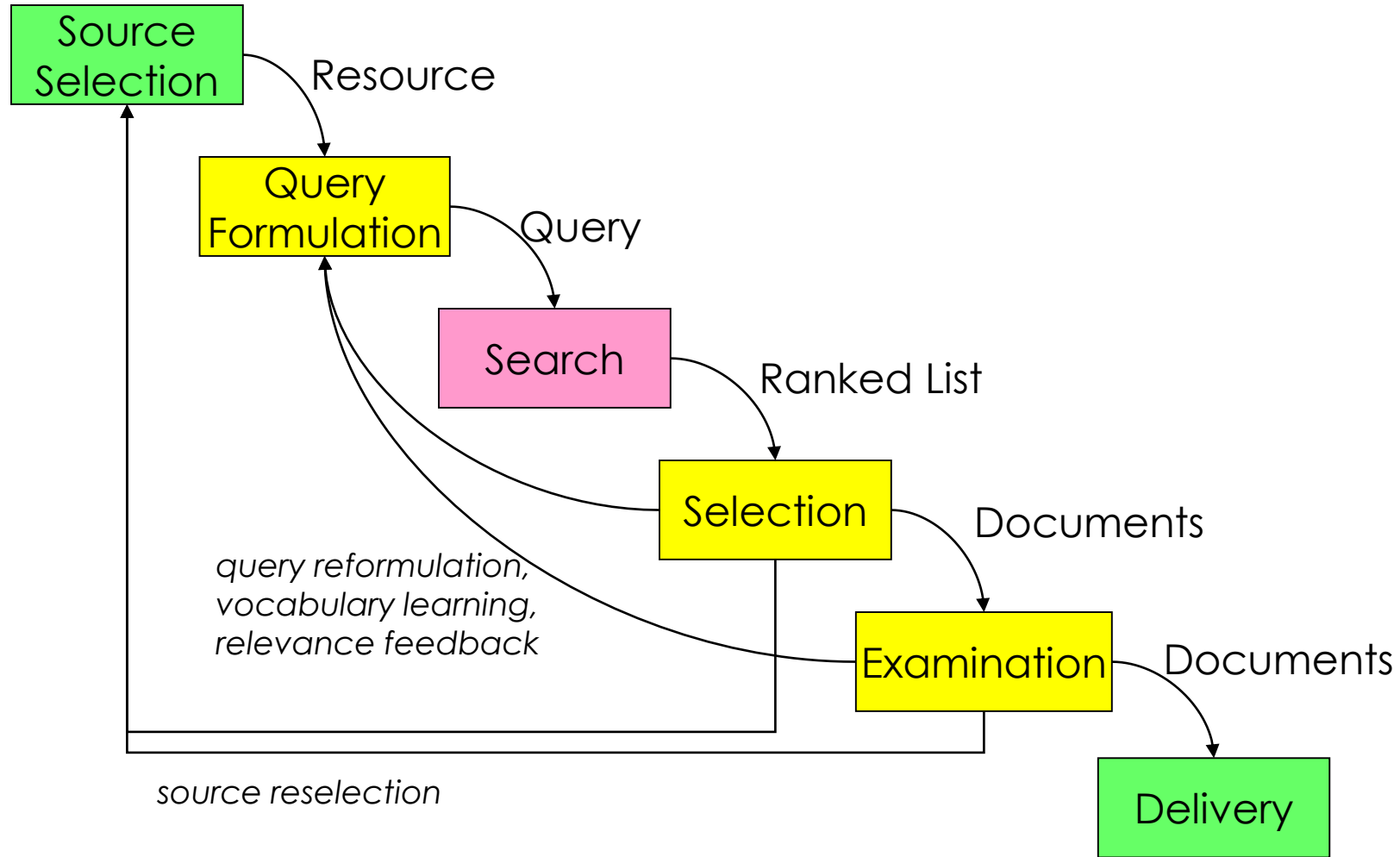
# Outline

- Information Retrieval (IR) vs. Information Extraction (IE)
  - Traditional IR
  - Web IR
  - IE
  - Non-traditional IE
    - Question Answering
    - Structured Summarization

# Information Retrieval

- Traditional Information Retrieval (IR)
  - User has an "information need"
  - User formulates query to retrieval system
  - Query is used to return matching documents

# The Information Retrieval Cycle



# IR Test Collections

- Three components of a test collection:
  - Collection of documents (corpus)
  - Set of information needs (topics)
  - Sets of documents that satisfy the information needs (relevance judgments)
- Metrics for assessing “performance”
  - Precision
  - Recall
  - Other measures derived therefrom (e.g., F1)

# Where do they come from?

- TREC = Text REtrieval Conferences
  - Series of annual evaluations, started in 1992
  - Organized into “tracks”
- Test collections are formed by “pooling”
  - Gather results from all participants
  - Corpus/topics/judgments can be reused

# Information Retrieval (IR)

- IMPORTANT ASSUMPTION: can substitute “document” for “information”
- IR systems
  - Use statistical methods
  - Rely on frequency of words in query, document, collection
  - Retrieve complete documents
  - Return ranked lists of “hits” based on relevance
- Limitations
  - Answers information need indirectly
  - Does not attempt to understand the “meaning” of user’s query or documents in the collection

# Web Retrieval

- Traditional IR came out of the library sciences
- Web search engines aren't only used like this
- Broder (2002) defined a taxonomy of web search engine requests
  - Informational (traditional IR)
    - When was Martin Luther King, Jr. assassinated?
    - Tourist attractions in Munich
  - Navigational (usually, want a website)
    - Deutsche Bahn
    - CIS, Uni Muenchen
  - Transactional (want to do something)
    - Buy Lady Gaga Pokerface mp3
    - Download Lady Gaga Pokerface (not that I am saying you would do this, for reasons of legality, or taste for that matter)
    - Order new Harry Potter book

# Web Retrieval

- Jansen et al (2007) studied 1.5 M queries

Type	Percentage of All Queries
Informational	81%
Navigational	10%
Transactional	9%

- Note that this probably doesn't capture the original intent well
  - Informational may often require extensive reformulation of queries



# Information Extraction (IE)

- Information Extraction is very different from Information Retrieval
  - Convert documents to zero or more database entries
  - Usually process entire corpus
- Once you have the database
  - Analyst can do further manual analysis
  - Automatic analysis ("data mining")
  - Can also be presented to end-user in a specialized browsing or search interface
    - For instance, concert listings crawled from music club websites (Tourfilter, Songkick, etc)

# Information Extraction (IE)

- IE systems
  - Identify documents of a specific type
  - Extract information according to pre-defined templates
  - Place the information into frame-like database records

Weather disaster:	Type	Damage
	Date	Deaths
	Location	...

- Templates = sort of like pre-defined questions
- Extracted information = answers
- Limitations
  - Templates are domain dependent and not easily portable
  - One size does not fit all!

# Question answering

- Question answering can be loosely viewed as "just-in-time" Information Extraction
- Some question types are easy to think of as IE templates, but some are not

	Who discovered Oxygen?
"Factoid"	When did Hawaii become a state?
	Where is Ayer's Rock located?
	What team won the World Series in 1992?
	What countries export oil?
"List"	Name U.S. cities that have a "Shubert" theater.
	Who is Aaron Copland?
"Definition"	What is a quasar?

# An Example

Who won the Nobel Peace Prize in 1991?

But many foreign investors remain sceptical, and western governments are withholding aid because of the Slorc's dismal human rights record and the continued detention of Ms Aung San Suu Kyi, the opposition leader who **won the Nobel Peace Prize in 1991**.

The military junta took power in 1988 as pro-democracy demonstrations were sweeping the country. It held elections in 1990, but has ignored their result. It has kept the **1991 Nobel peace prize winner**, Aung San Suu Kyi - leader of the opposition party which won a landslide victory in the poll - under house arrest since July 1989.

The regime, which is also engaged in a battle with insurgents near its eastern border with Thailand, ignored a 1990 election victory by an opposition party and is detaining its leader, Ms Aung San Suu Kyi, who was awarded the **1991 Nobel Peace Prize**. According to the British Red Cross, 5,000 or more refugees, mainly the elderly and women and children, are crossing into Bangladesh each day.

# Central Idea of Factoid QA

- Determine the semantic type of the expected answer

“Who won the Nobel Peace Prize in 1991?” is looking for a PERSON

- Retrieve documents that have keywords from the question

Retrieve documents that have the keywords “won”, “Nobel Peace Prize”, and “1991”

- Look for named-entities of the proper type near keywords

Look for a PERSON near the keywords “won”, “Nobel Peace Prize”, and “1991”

# Structured Summarization

- Typical automatic summarization task is to take as input an article, and return a short text summary
  - Good systems often just choose sentences (reformulating sentences is difficult)
- A structured summarization task might be to take a company website, say, [www.inxight.com](http://www.inxight.com), and return something like this:

Company Name: Inxight

Founded: 1997

History: Spun out from Xerox PARC Business

Focus: Information Discovery from Unstructured Data Sources

Industry Focus: Enterprise, Government, Publishing, Pharma/Life Sciences, Financial Services, OEM

Solutions: Based on 20+ years of research at Xerox PARC

Customers: 300 global 2000 customers

Patents: 70 in information visualization, natural language processing, information retrieval

Headquarters: Sunnyvale, CA

Offices: Sunnyvale, Minneapolis, New York, Washington DC, London, Munich, Boston, Boulder, Antwerp

# Non-traditional IE

- We discussed two other interesting IE scenarios
  - Question answering
  - Structured summarization
- There are many more
  - For instance, think about how information from IE can be used to improve Google queries and results
    - As discussed in Sarawagi

# Outline

- IE Scenario
- Source selection
- Tokenization and normalization
- Extraction of entities in closed and regular sets
  - e.g., dates, country names



# Finding the Sources



How can we find the documents to extract information from?

- The document collection can be given a priori (**Closed** Information Extraction)  
e.g., a specific document, all files on my computer, ...
- We can aim to extract information from the entire Web (**Open** Information Extraction)  
For this, we need to crawl the Web
- The system can find by itself the source documents  
e.g., by using an Internet search engine such as Google

# Scripts

Elvis Presley was a rock star.

(Latin script)

猫王是摇滚明星

(Chinese script,  
“simplified”)

אלביס היה כוכב רוק

(Hebrew)

وكان ألفيس بريسلي نجم الروك

(Arabic)

록 스타 엘비스 프레슬리

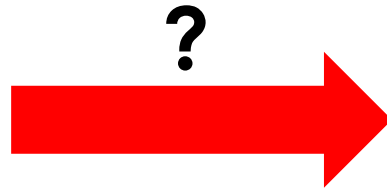
(Korean script)

Elvis Presley ถูกดาวร็อก

(Thai script)

# Char Encoding: ASCII

100,000 different  
characters  
from 90 scripts



One byte with 8 bits  
per character  
(can store numbers 0-255)

How can we encode so many characters in 8 bits?

- Ignore all non-English characters (**ASCII standard**)

26 letters + 26 lowercase letters + punctuation  $\approx$  100 chars

Encode them as follows:

A=65,

B=66,

C=67,

...

Disadvantage: Works only for English

# Char Encoding: Code Pages

- For each script, develop a different mapping (a **code-page**)

Hebrew code page: ....., 226=כ,...

Western code page: ....., 226=à,...

Greek code page: ....., 226=α, ...

(most code pages map characters 0-127 like ASCII)

Disadvantages:

- We need to know the right code page
- We cannot mix scripts

# Char Encoding: HTML

- Invent special sequences for special characters (e.g., **HTML entities**)

&egrave; = è, ...

Disadvantage: Very clumsy for non-English documents

# Char Encoding: Unicode

- Use 4 bytes per character (**Unicode**)

...65=A, 66=B, ..., 1001=α, ..., 2001=ㄹ|

Disadvantage: Takes 4 times as much space as ASCII

# Char Encoding: UTF-8

- Compress 4 bytes Unicode into 1-4 bytes (**UTF-8**)

Characters **0 to 0x7F** in Unicode:

Latin alphabet, punctuation and numbers

Encode them as follows:

**0xxxxxxx**

(i.e., put them into a byte, fill up the 7 least significant bits)

**A = 0x41 = 1000001**



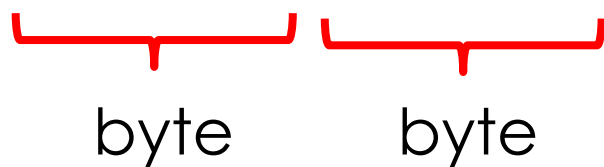
**01000001**

Advantage: An UTF-8 byte that represents such a character is equal to the ASCII byte that represents this character.

# Char Encoding: UTF-8

Characters  $0x80-0x7FF$  in Unicode (11 bits):  
Greek, Arabic, Hebrew, etc.

Encode as follows:

$110xxxxx$   $10xxxxxx$   
  
byte      byte

$\zeta = 0xE7 = 00011100111$



$11000011$   $10100111$

f                      a                      ζ                                      a      d      e

0x66                      0x61                      0xE7                                      0x61                      ....

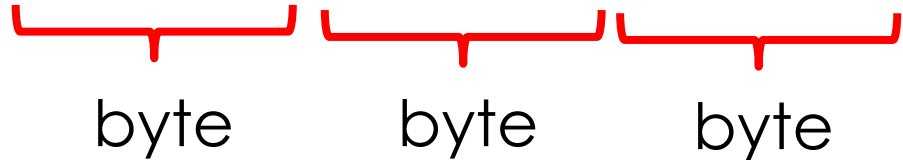
01100110      01100001      11000011      10100111      01100001



# Char Encoding: UTF-8

Characters **0x800-0xFFFF** in Unicode (16 bits):  
mainly Chinese

Encode as follows:

1110**xxxx** 10**xxxxxxx** 10**xxxxxxx**  
  
byte          byte          byte

# Char Encoding: UTF-8

Decoding (mapping a sequence of bytes to characters):

- If the byte starts with `0xxxxxxx`  
=> it's a "normal" character 00-0x7F
- If the byte starts with `110xxxxx`  
=> it's an "extended" character 0x80 - 0x77F  
one byte will follow
- If the byte starts with `1110xxxx`  
=> it's a "Chinese" character, two bytes follow
- If the byte starts with `10xxxxxx`  
=> it's a follower byte, not valid!

01100110    01100001    11000011    10100111    01100001  
f            a            ç            a

# Char Encoding: UTF-8

**UTF-8** is a way to encode all Unicode characters into a variable sequence of 1-4 bytes

Advantages:

- common Western characters require only 1 byte (😊)
- backwards compatibility with ASCII
- stream readability (follower bytes cannot be confused with marker bytes)
- sorting compliance

In the following, we will assume that the document is a sequence of characters, without worrying about encoding

# Language detection

How can we find out the language of a document?

Elvis Presley ist einer der größten Rockstars aller Zeiten.

Different techniques:

- Watch for certain characters or scripts (umlauts, Chinese characters etc.)  
But: These are not always specific, Italian similar to Spanish
- Use the meta-information associated with a Web page  
But: This is usually not very reliable
- Use a dictionary  
But: It is costly to maintain and scan a dictionary for thousands of languages

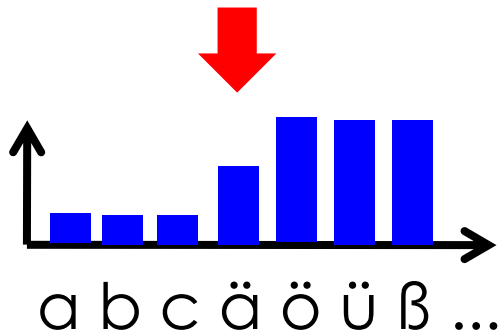
# Language detection

**Histogram technique** for language detection:

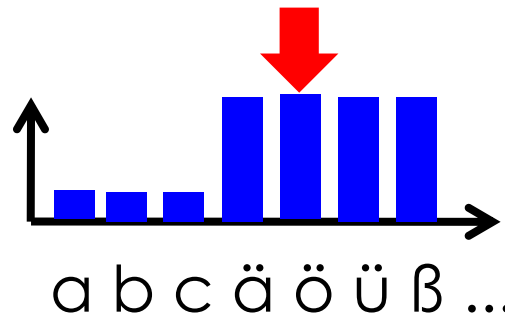
Count how often each character appears in the text.

Document:

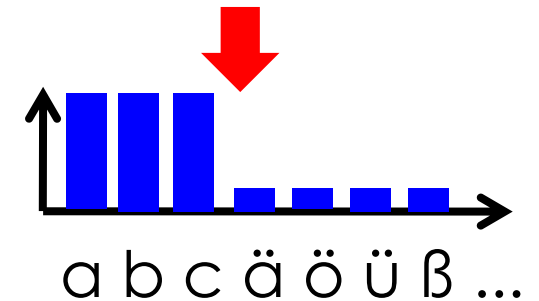
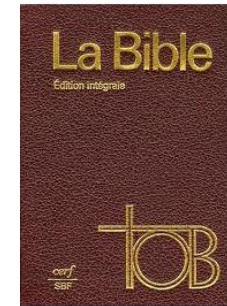
Elvis Presley ist  
...



German corpus:



French corpus:



*similar* (between German corpus and document)

*not very similar* (between French corpus and document)

Then compare to the counts on standard corpora.

# Sources: Structured

Name	Number
D. Johnson	30714
J. Smith	20934
S. Shenker	20259
Y. Wang	19471
J. Lee	18969
A. Gupta	18884
R. Rivest	18038

**Information  
Extraction**



Name	Citations
D. Johnson	30714
J. Smith	20937
...	...

File formats:

- TSV file (values separated by tabulator)
- CSV (values separated by comma)

# Sources: Semi-Structured

```
<catalog>
```

```
  <cd>
```

```
    <title>
```

```
      Empire Burlesque
```

```
    </title>
```

```
    <artist>
```

```
      <firstName>
```

```
        Bob
```

```
      </firstName>
```

```
      <lastName>
```

```
        Dylan
```

```
      </lastName>
```

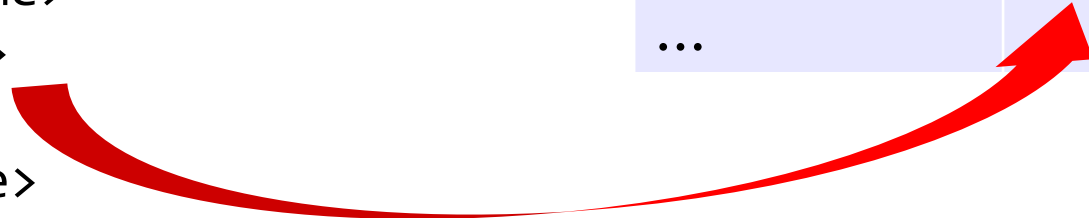
```
    </artist>
```

```
  </cd>
```

**Information  
Extraction**



Title	Artist
Empire Burlesque	Bob Dylan
...	



File formats:

- XML file (Extensible Markup Language)
- YAML (Yaml Ain't a Markup Language)

# Sources: Semi-Structured

Release Date (Year/M./Day)	Title	Num. Tracks
2008-11-24	Miles Away	7
2008-01-01	Hard Candy Cover	15
2008-01-01	Hard Candy Cover	12
2008-01-01	4 Minutes (4-Track Maxi-Single)	4
2008-01-01	4 Minutes (Single) Cover	1

<table>

<tr>

<td> 2008-11-24

<td> Miles away

<td> 7

</tr>

...

Information  
Extraction



Title	Date
Miles away	2008-11-24
...	...

File formats:

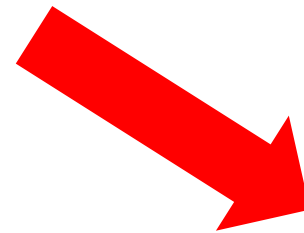
- HTML file with table (Hypertext Markup Lang.)
- Wiki file with table (later in this class)



# Sources: “Unstructured”

Founded in 1215 as a colony of Genoa, Monaco has been ruled by the House of Grimaldi since 1297, except when under French control from 1789 to 1814.

Designated as a protectorate of Sardinia from 1815 until 1860 by the Treaty of Vienna, Monaco's sovereignty ...







**Information  
Extraction**

Event	Date
Foundation	1215
...	...

File formats:

- HTML file
- text file
- word processing document

# Sources: Mixed

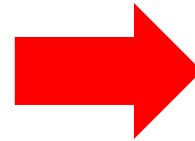
<b>Barto, Andrew G.</b>	(413) 545-2109	<a href="mailto:barto@cs.umass.edu">barto@cs.umass.edu</a>	CS276
Professor. Computational neuroscience, reinforcement learning, adaptive motor control, artificial neural networks, adaptive and learning control, motor development.			 
<b>Berger, Emery D.</b>	(413) 577-4211	<a href="mailto:emery@cs.umass.edu">emery@cs.umass.edu</a>	CS344
Assistant Professor.			 

<table>

<tr>

<td> Professor.  
Computational  
Neuroscience,  
...

Information  
Extraction



Name	Title
Barte	Professor
...	...

...

Different IE approaches work with different types of sources

# Source Selection Summary

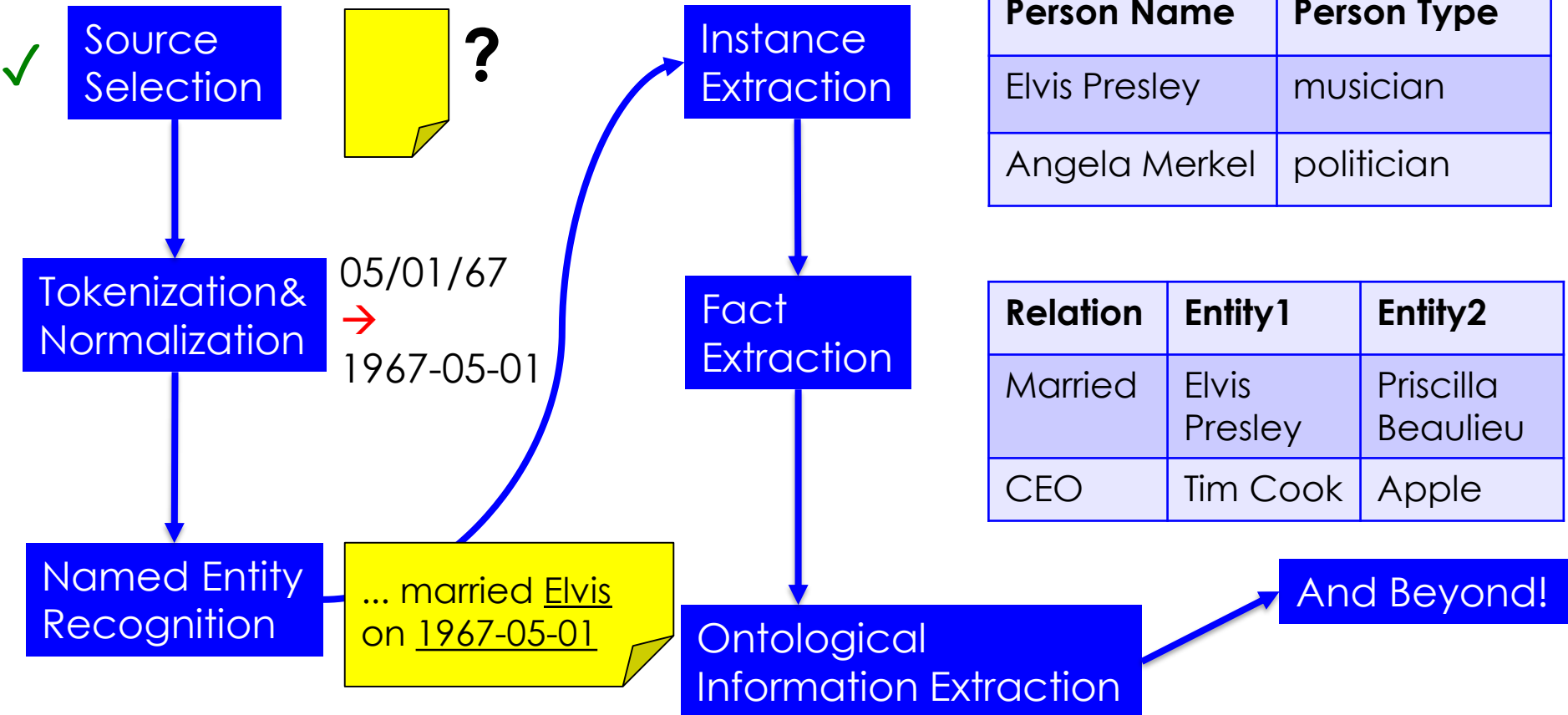
We can extract from the entire Web, or from certain Internet domains, thematic domains or files.

We have to deal with character encodings (ASCII, Code Pages, UTF-8,...) and detect the language

Our documents may be structured, semi-structured or unstructured.

# Information Extraction

**Information Extraction (IE)** is the process of extracting **structured information** from unstructured machine-readable documents



# Tokenization

**Tokenization** is the process of splitting a text into tokens.

A **token** is

- a word
- a punctuation symbol
- a url
- a number
- a date
- or any other sequence of characters regarded as a unit

In|2011|,| President|Sarkozy|spoke|this|sample|sentence|. .

# Tokenization Challenges

In | 2011 | , | President | Sarkozy | spoke | this | sample | sentence | .

Challenges:

- In some languages (Chinese, Japanese), words are not separated by white spaces
- We have to deal consistently with URLs, acronyms, etc.  
<http://example.com>, 2010-09-24, U.S.A.
- We have to deal consistently with compound words  
[hostname](#), [host-name](#), [host name](#)

⇒ Solution depends on the language and the domain.

Naive solution: split by white spaces and punctuation

# Normalization: Strings

Problem: We might extract strings that differ only slightly and mean the same thing.

Elvis Presley	singer
ELVIS PRESLEY	singer

Solution: **Normalize** strings, i.e., convert strings that mean the same to one common form:

- **Lowercasing**, i.e., converting all characters to lower case
- **Removing accents and umlauts**  
résumé → resume, Universität → Universitaet
- **Normalizing abbreviations**  
U.S.A. → USA, US → USA

# Normalization: Literals

Problem: We might extract different **literals** (numbers, dates, etc.) that mean the same.

Elvis Presley	1935-01-08
Elvis Presley	08/01/35

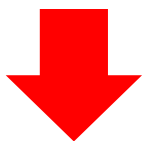
Solution: **Normalize** the literals, i.e., convert equivalent literals to one standard form:

08/01/35

01/08/35

8<sup>th</sup> Jan. 1935

January 8<sup>th</sup>, 1935



1935-01-08

1.67m

1.67 meters

167 cm

6 feet 5 inches



1.67m



# Normalization

Conceptually, normalization groups tokens into equivalence classes and chooses one representative for each class.

resume

résumé,  
resume,  
Resume

1935-01-08

8<sup>th</sup> Jan 1935,  
01/08/1935

Take care not to normalize too aggressively:

bush

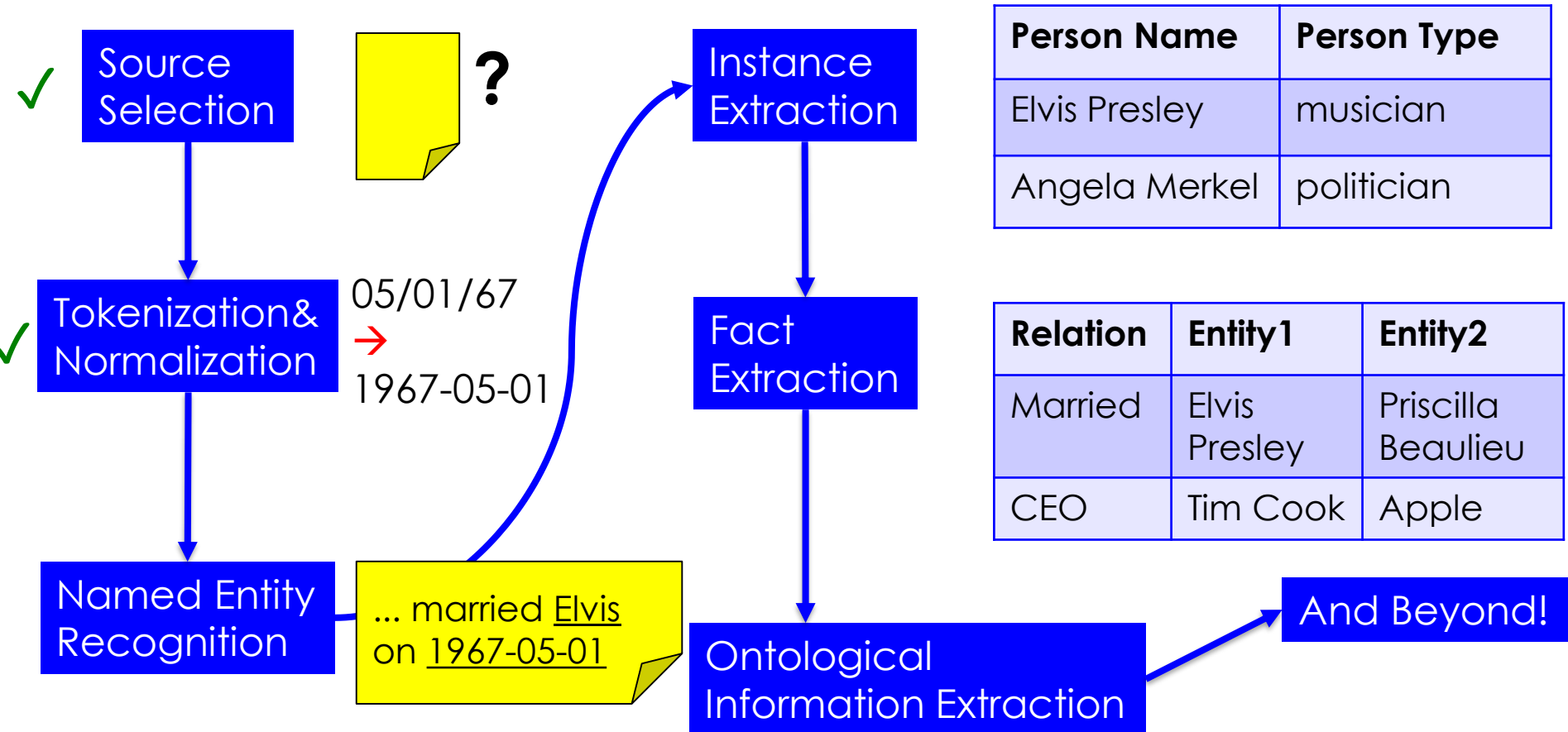


# Caveats

- Even the "simple" task of normalization can be difficult
  - Sometimes you require information about the semantic class
  - If the sentence is "Bush is characteristic.", is it bush or Bush?
    - Hint, you need at least the previous sentence...

# Information Extraction

**Information Extraction (IE)** is the process of extracting **structured information** from unstructured machine-readable documents



# Named Entity Recognition

**Named Entity Recognition** (NER) is the process of finding entities (people, cities, organizations, dates, ...) in a text.

Elvis Presley was born in 1935 in East Tupelo, Mississippi.



# Closed Set Extraction

If we have an exhaustive set of the entities we want to extract, we can use **closed set extraction**:

Comparing every string in the text to every string in the set.

... in Tupelo, Mississippi, but ...

States of the USA  
{ Texas, Mississippi,... }

... while Germany and France were opposed to a 3<sup>rd</sup> World War, ...

Countries of the World (?)  
{ France, Germany, USA,... }

May not always be trivial...

... was a great fan of France Gall, whose songs...

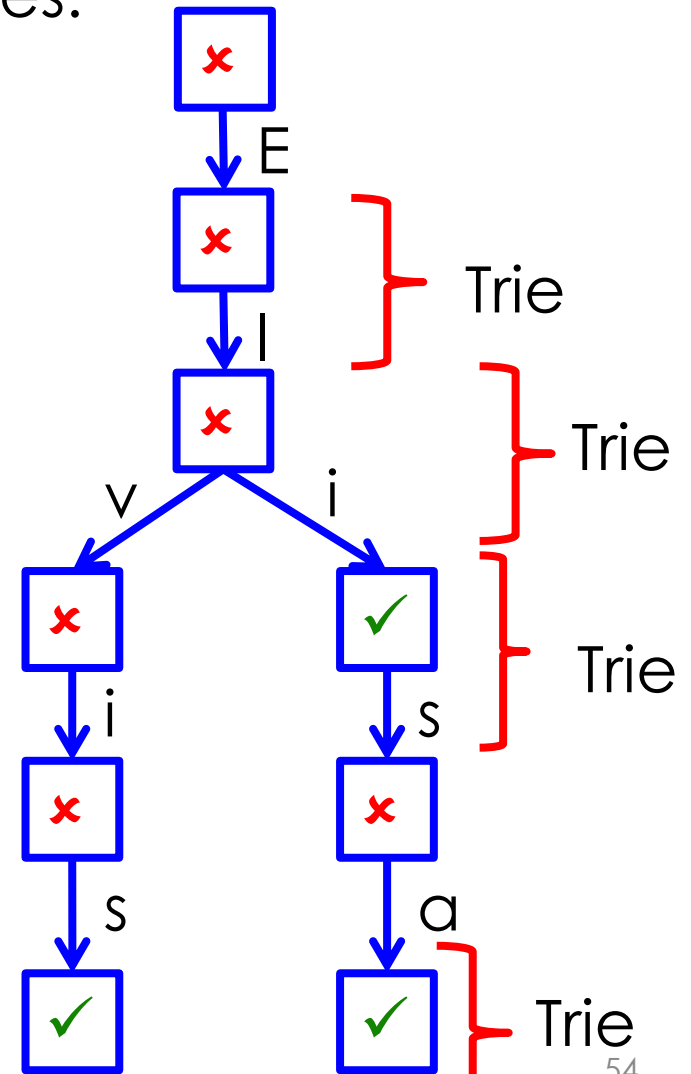
How can we do that efficiently?

# Tries

A **trie** is pair of a boolean truth value, and a function from characters to tries.

Example: A trie containing “Elvis”, “Elisa” and “Eli”

A trie contains a string, if the string denotes a path from the root to a node marked with TRUE (✓)



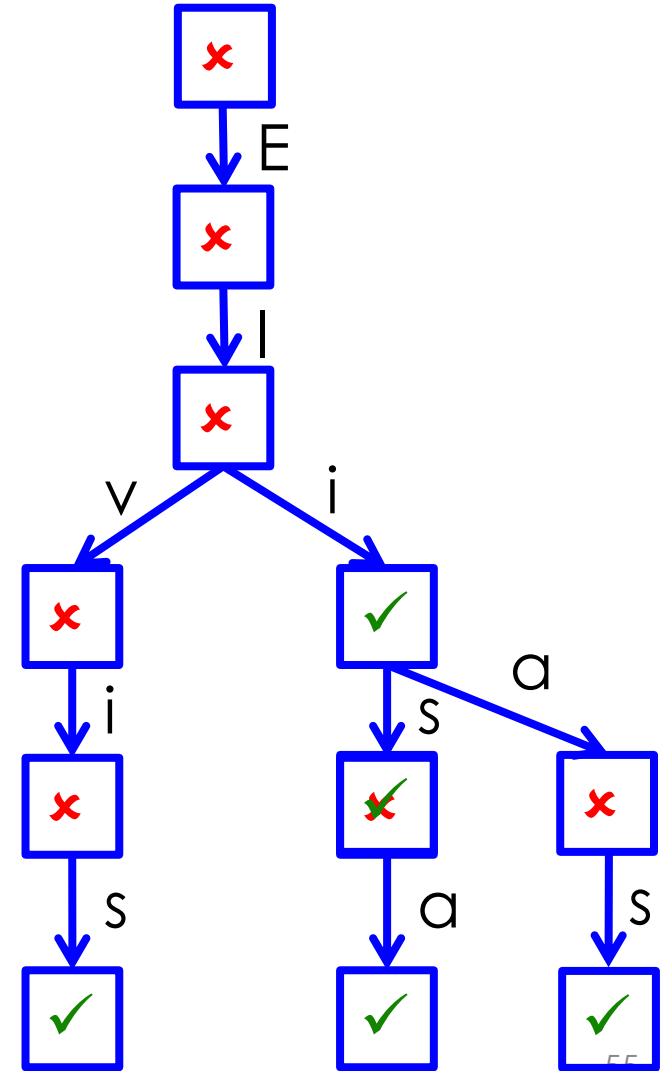
# Adding Values to Tries

Example: Adding "Elis"

Switch the sub-trie to TRUE (✓)

Example: Adding "Elias"

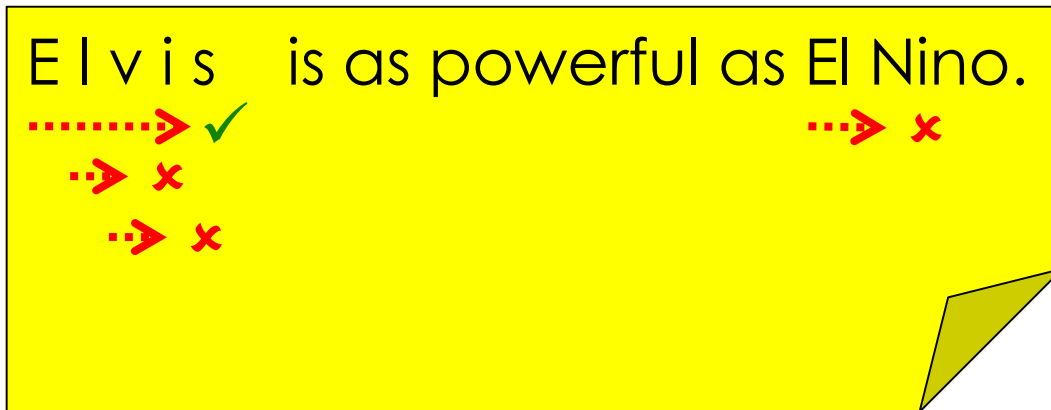
Add the corresponding sub-trie



# Parsing with Tries

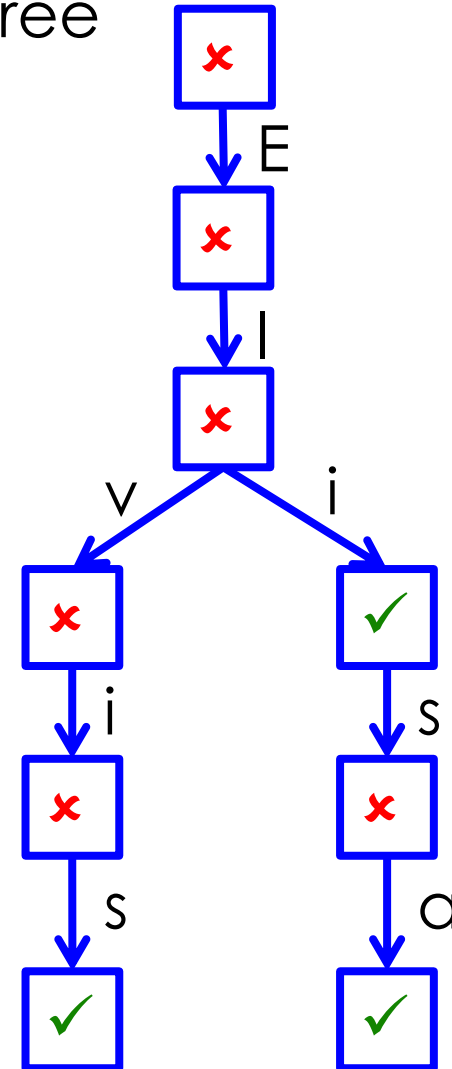
For every character in the text,

- advance as far as possible in the tree
- report match if you meet a node marked with TRUE (✓)



=> found Elvis

Time:  $O(\text{textLength} * \text{longestEntity})$





# NER: Patterns

If the entities follow a certain pattern, we can use **patterns**

... was born in 1935. His mother...  
... started playing guitar in 1937, when...  
... had his first concert in 1939, although...

Years  
(4 digit numbers)

Office: 01 23 45 67 89  
Mobile: 06 19 35 01 08  
Home: 09 77 12 94 65

Phone numbers  
(groups of digits)

# Patterns

A **pattern** is a string that generalizes a set of strings.

sequences of the letter 'a'

$a^+$

a    aa    aaaaaa  
          aaaa  
aaaaaaa

'a', followed by 'b's

$ab^+$

abbbbbbb    abbbb  
              ab    abbb  
                          abb

digits

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

0 9 1 6 2  
8 3 5 7 4

sequence of digits

$(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)^+$

987    6543    5321  
          5643

=> Let's find a systematic way of expressing patterns

# Regular Expressions

A **regular expression** (regex) over a set of symbols  $\Sigma$  is:

1. the empty string
2. or the string consisting of an element of  $\Sigma$   
(a single character)
3. or the string  $AB$  where  $A$  and  $B$  are regular expressions  
(**concatenation**)
4. or a string of the form  $(A | B)$ ,  
where  $A$  and  $B$  are regular expressions (**alternation**)
5. or a string of the form  $(A)^*$ ,  
where  $A$  is a regular expression (**Kleene star**)

For example, with  $\Sigma=\{a,b\}$ , the following strings are regular expressions:

a

b

ab

aba

(a | b)

# Regular Expression Matching

## Matching

- a string **matches** a regex of a single character if the string consists of just that character

a

b

← regular expression

a

b

← matching string

- a string matches a regular expression of the form  $(A)^*$  if it consists of zero or more parts that match  $A$

(a)\*

← regular expression

aaa aaaaa

aaaaa

← matching strings

# Regular Expression Matching

## Matching

- a string matches a regex of the form  $(A | B)$  if it matches either A or B

$(a | b)$

b a

$(a | (b)^*)$

bbbb bb  
a

← regular expression

← matching strings

- a string matches a regular expression of the form AB if it consists of two parts, where the first part matches A and the second part matches B

$ab$

ab

$b(a)^*$

baa  
b baaaa

← regular expression

← matching strings

# Additional Regexes

Given an ordered set of symbols  $\Sigma$ , we define

- $[x-y]$  for two symbols  $x$  and  $y$ ,  $x < y$ , to be the alternation  $x | \dots | y$  (meaning: any of the symbols in the range)  
 $[0-9] = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$
- $A^+$  for a regex  $A$  to be  $A(A)^*$  (meaning: one or more  $A$ 's)  
 $[0-9]^+ = [0-9][0-9]^*$
- $A\{x,y\}$  for a regex  $A$  and integers  $x < y$  to be  $A \dots A | A \dots A | A \dots A | \dots | A \dots A$  (meaning:  $x$  to  $y$   $A$ 's)  
 $f\{4,6\} = ffff | fffff | fffffff$
- $A?$  for a regex  $A$  to be  $( | A)$  (meaning: an optional  $A$ )  
 $ab? = a( | b)$
- $.$  to be an arbitrary symbol from  $\Sigma$

# Things that are easy to express

$A \mid B$	Either A or B	(Use a backslash for the character itself, e.g., $\backslash+$ for a plus)
$A^*$	Zero+ occurrences of A	
$A^+$	One+ occurrences of A	
$A\{x,y\}$	x to y occurrences of A	
$A?$	an optional A	
$[a-z]$	One of the characters in the range	
$.$	An arbitrary symbol	

A digit

A digit or a letter

A sequence of 8 digits

5 pairs of digits, separated by space

HTML tags

Person names:

Dr. Elvis Presley

Prof. Dr. Elvis Presley

# Names & Groups in Regexes

When using regular expressions in a program, it is common to **name** them:

```
String digits="[0-9]+";  
String separator="( |-)";  
String pattern=digits+separator+digits;
```

Parts of a regular expression can be singled out by bracketed **groups**:

```
String input="The cat caught the mouse."
```

```
String pattern="The ([a-z]+) caught the ([a-z]+)\\.\\."
```

 first group: "cat"  
second group: "mouse"



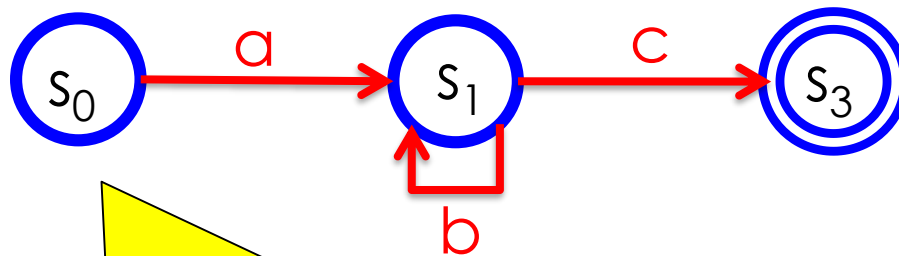
# Finite State Machines

A regex can be matched efficiently by a Finite State Machine (Finite State Automaton, FSA, FSM)

A **FSM** is a quintuple of

- A set  $\Sigma$  of symbols (the **alphabet**)
- A set  $S$  of **states**
- An **initial state**,  $s_0 \in S$
- A **state transition function**  $\delta: S \times \Sigma \rightarrow S$
- A **set of accepting states**  $F \subset S$

Regex:  $ab^*c$



Accepting states usually depicted with double ring.

Implicitly: All unmentioned inputs go to some artificial failure state

# Finite State Machines

A FSM **accepts** an input string, if there exists a sequence of states, such that

- it starts with the start state
- it ends with an accepting state
- the  $i$ -th state,  $s_i$ , is followed by the state  $\delta(s_i, \text{input.charAt}(i))$

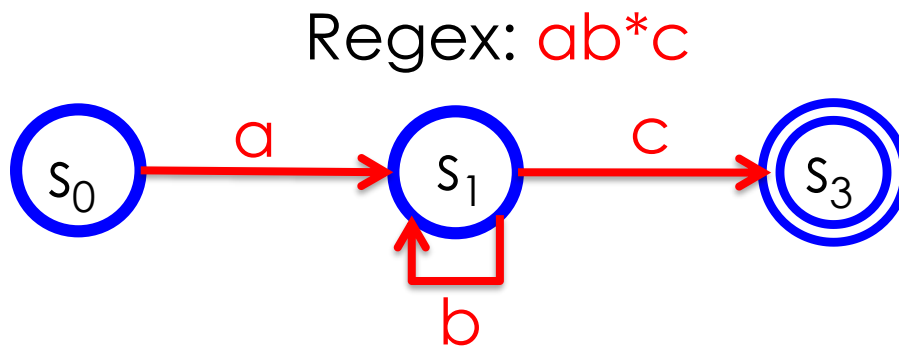
Sample inputs:

abbbc

ac

aabbbc

elvis



# Regular Expressions Summary

## Regular expressions

- can express a wide range of patterns
- can be matched efficiently
- are employed in a wide variety of applications (e.g., in text editors, NER systems, normalization, UNIX grep tool etc.)

## Input:

- Manual design of the regex

## Condition:

- Entities follow a pattern

# Entity matching techniques

- A last word for today on Entity Matching
- **Rule-based techniques** are still heavily used heavily in (older) industrial applications
  - The patterns sometimes don't capture an entity when they should
    - But the emphasis in industry is often on being right when you do match
    - Not matching at all is often considered better (in industry) when the match is doubtful
  - With rule-based it is easy to understand what is happening
    - Easy to make changes so that a particular example is extracted correctly
- However, **statistical techniques** have recently become much more popular
  - E.g., Google
  - Emphasis is much more on higher coverage and noisier input
  - We will discuss both in this class
    - But with a stronger emphasis on statistical techniques and hybrid techniques (combining rules with statistics)
- Don't forget to read Sarawagi on rule-based NER!

- Slide sources
  - Slides today were original and from a variety of sources (see bottom right of each slide)
  - I'd particularly like to mention Jimmy Lin, Maryland and Fabian Suchanek, Télécom ParisTech

- Thank you for your attention!



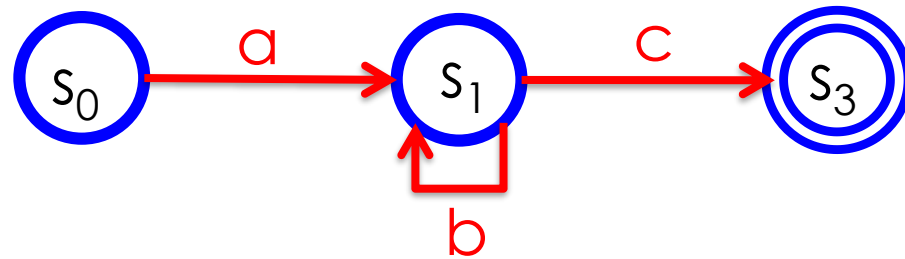
- NOT CURRENTLY USED



# Finite State Machines

Example (from previous slide):

Regex:  $ab^*c$



Exercise:

*Draw a FSM that can recognize comma-separated sequences of the words “Elvis” and “Lisa”:*

*Elvis, Elvis, Elvis*

*Lisa, Elvis, Lisa, Elvis*

*Lisa, Lisa, Elvis*

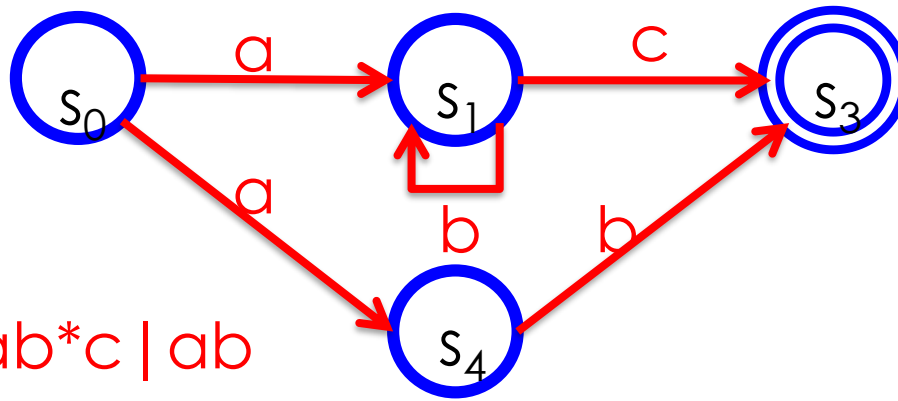
*...*

# Non-Deterministic FSM

A **non-deterministic FSM** has a transition function that maps to a **set of states**.

A FSM **accepts** an input string, if there exists a sequence of states, such that

- it starts with the start state
- it ends with an accepting state
- the  $i$ -th state,  $s_i$ , is followed by a state in the set  $\delta(s_i, \text{input.charAt}(i))$



Regex:  $ab^*c \mid ab$

Sample inputs:

abbbc

ab

abc

elvis