ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

Lecture 5: Web mining and Information retrieval for knowledge graph building

Yiru Zhang

Mathematics for machine learning

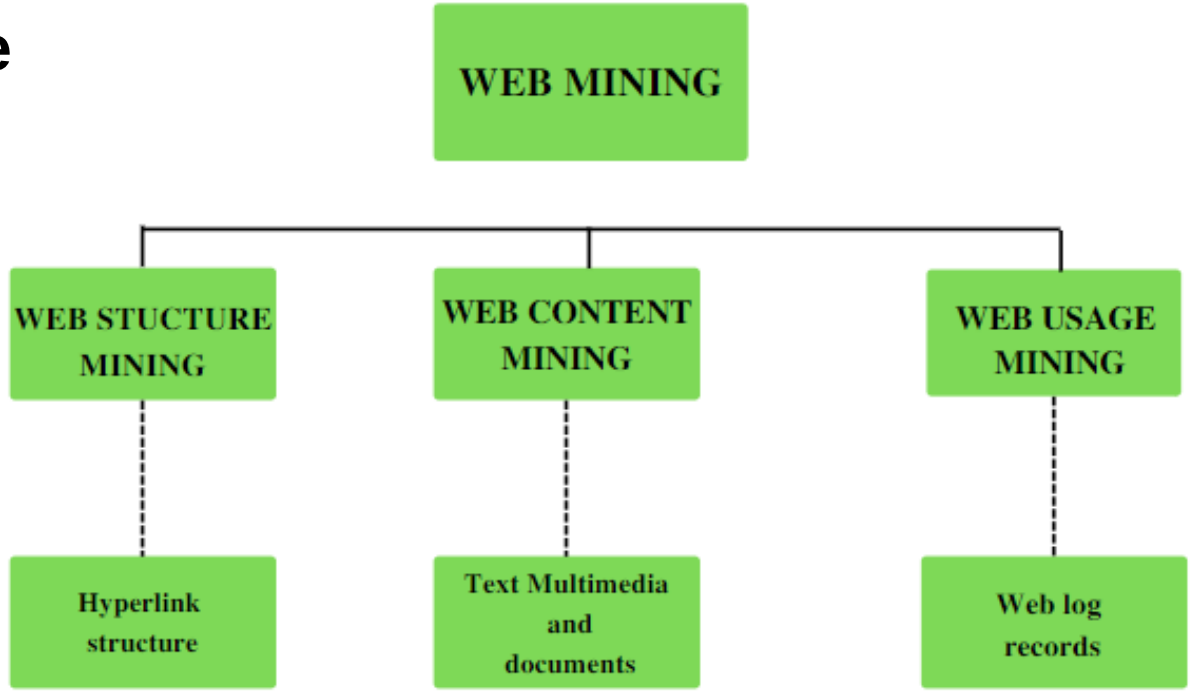
Département d'Informatique

Ecole Supérieure d'Ingénieurs Léonard de Vinci

DE VINCI
RESEARCH CENTER

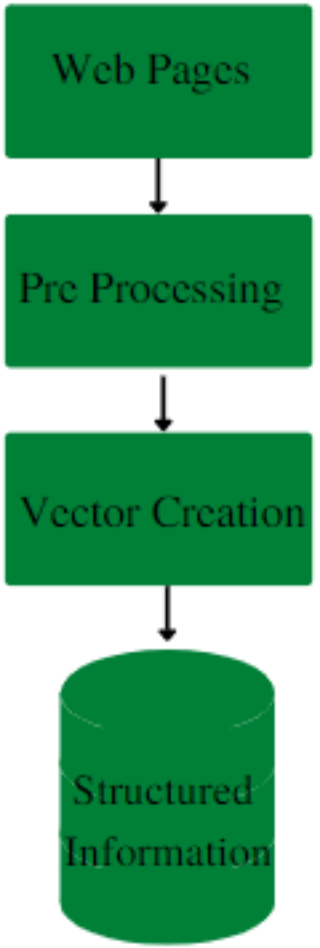
What Is Web Mining?

- Extract useful knowledge from web data
- It targets three main data types:
  - **Web Content**
  - **Web Structure**
  - **Web Usage**



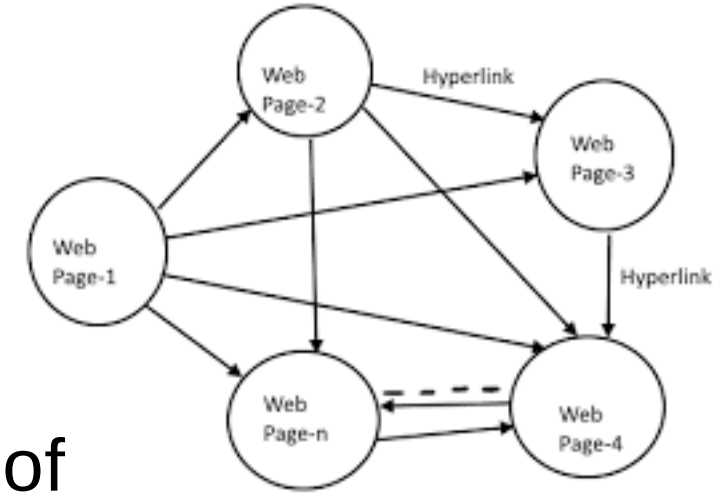
Web Content Mining

- Extracting meaningful information from web pages.
  - Text, Image, Audio, Video
- Difficulties: Web data are generally **semi-structured and/or unstructured**. We need to convert them into structured data. Mostly **Text Mining** techniques(we will learn some solutions later)
- Examples:
  - Classifying the web documents into categories.
  - Identify topics of web documents.
  - Finding similar web pages across the different web servers.
  - Applications related to relevance.



ESILV

Web Structure Mining

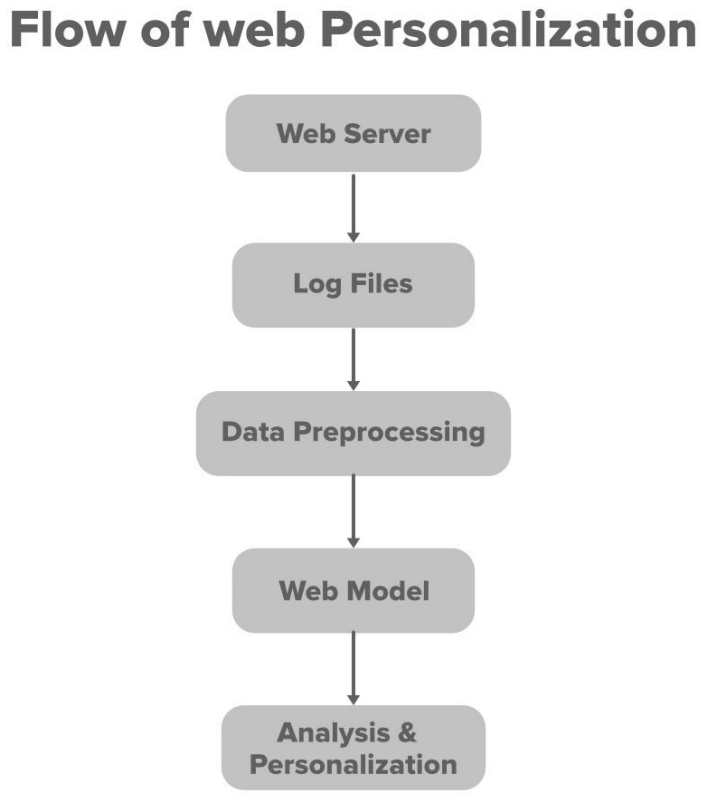


- Discovering structure information from the web. Usually, the structure of the web graph consists of web pages as **nodes**, and hyperlinks as **edges** connecting related pages.
- Mostly **Graph Mining** techniques (You will learn on the 5th year)
- Examples:
  - Information retrieval in social networks (graph)
  - To find out the relevance of each web page.
  - Used in Search engines to find the relevant information.

ESILV

Web Usage Mining

- Identifying or discovering interesting **usage patterns** from large data sets.
- These patterns may help understanding the behaviours or **profile** of the users.
- Usually, user access data is in form of logs (structured data).
- Applications:
  - Privatization of we content (recommendation system)
  - Pre-recovery (to improve the performance of we servers)



ESILV

Challenges in web mining

- Deluge of data
  - Large-scale data with unknown distributions
  - Often little ratio of data is available and/or useful
  - Sampling is hard
  - Data is noisy
  - Data is not always trusty
  - Heterogeneous data and sources
- Diversity of user profiles
  - Paradox of data availability and privacy for personalization purposes
  - Heterogeneity of connections
- Evaluation Dilemma
  - Building ground truth is costly

ESILV

What will we learn in this course?

- Web structure mining? No
  - Mostly graph mining techniques, you will have a such module in A5
- Web usage mining? No
  - Mostly applications on database management and recommendation system building.
- Web content mining? Yes
  - We will learn how to build knowledge base from flat text.

ESILV

Information Extraction (IE) at a glance

- Identifying and contextualizing:
  - Entities
  - Locations
  - Dates
  - Events
  - Relations
  - Quantities
  - Etc.

© France Info Publié le 06/09/2022 13:26Mis à jour le 06/09/2022 13:38

Un pétrolier russe attend pour décharger sa cargaison dans le golfe de Mexique (Etats-Unis), le 03 juillet 2002. (POOL / BLOOMBERG NEWS / AFP)

Un embargo qui ne semble pas avoir asphyxié Moscou. La Russie a tiré 158 milliards d'euros de revenus des exportations de combustibles fossiles au cours des six premiers mois de la guerre en Ukraine. C'est le principal constat dressé par les experts du Centre de recherche sur l'énergie et l'air pur (CREA) dans un rapport *(en anglais)* publié mardi 6 septembre.

Au premier trimestre, l'Union européenne a importé pour près de 85 milliards d'euros de combustibles fossiles russes, faisant des Vingt-Sept les plus gros clients de la Russie, suivie de la Chine (34,9 milliards d'euros), de la Turquie (10,7 milliards d'euros), de l'Inde (6,6 milliards d'euros), du Japon (2,5 milliards d'euros), de l'Egypte (2,3 milliards d'euros, et la Corée du Sud (2 milliards d'euros).

Information Extraction (IE) at a glance

- Producing structured information

Author	*Marie-Francine Moens*
Title	*Information extraction, algorithms and propspects in a retrieval context*
Price	*158,34 Euro*
Date	December 29th *2009*

- Gathering information from multiple resources (pieces of texts, sentences, documents)

Equipe	*Paris Saint-Germain*
Nom	*Kylian Mbappé*
Naissance	*20/12/1998*
Chiffre symbolique	3

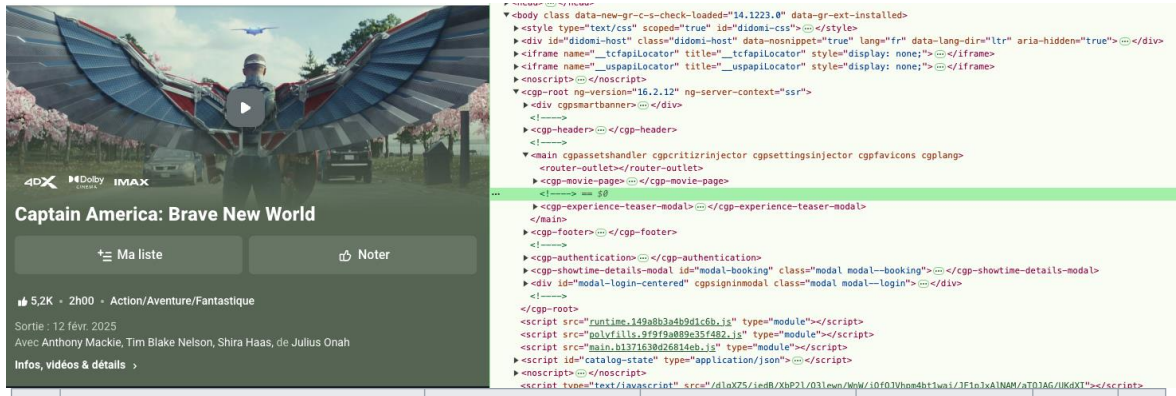
Information Extraction (IE) at a glance

- Identify relevant information from documents
  - Unstructured texts
    - Free form, common grammar and words, free language, subjective/objective
    - Could not be "naturally" indexed into a structured database
  - Semi-structured texts
    - Layout/template, text within a format
  - Tabular forms
- Objective: Convert into structured text (that machines understand)

The **Leonard de Vinci Engineering School** (ESILV) is an engineering school located in Paris, France. It is fully accredited to award the title of *ingénieur* by the French Commission of Engineering Titles. ESILV is part of the "Pôle universitaire Léonard-de-Vinci". The director of the school is Pascal Pinot.

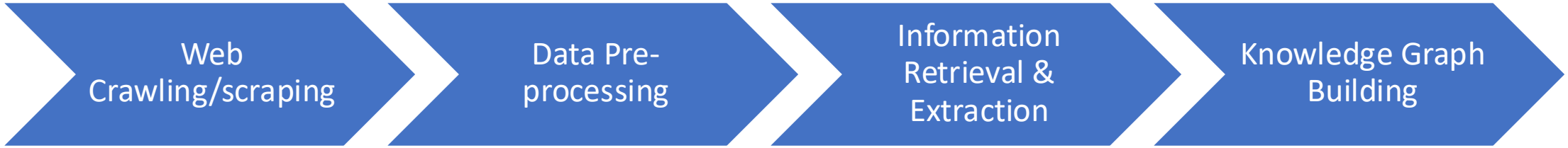
It was placed 10th among French engineering schools "Post-bac" in the 2010 *Le Point* rankings.[1]

Since 2022, the school has been the highest-ranked private school post-high school according to *Usine Nouvelle*, *L'Étudiant*, and *Le Figaro*. In 2024, ESILV was ranked 2nd by *L'Étudiant*, 3rd by *Le Figaro* in post-baccalaureate rankings, and 1st by *Usine Nouvelle* in the same category. Furthermore, in 2023, it was ranked 3rd in the



	Country / dependency	Total in km² (mi²)	Land in km² (mi²)	Water in km² (mi²)	% water	
–	Abkhazia	8,665 (3,346)				[bw]
40	Afghanistan	652,864 (252,072)	652,230 (251,830)	630 (240)	0.1	
–	Akrotiri and Dhekelia (UK)	254 (98)				[cr]
–	Åland (Finland)	1,581 (610)	1,553 (600)	28 (11)	1.8	[ce]
140	Albania	28,748 (11,100)	27,400 (10,600)	330 (130)	1.1	
10	Algeria	2,381,741 (919,595)	2,381,741 (919,595)	0	0	[m]
–	American Samoa (US)	199 (77)	199 (77)	0	0	

The Overall Pipeline



- To harvest raw content, including text, images, and metadata
- Applications
  - Search engine indexing,
  - competitive intelligence,
  - data aggregation,
  - monitoring online content.

- Clean, transform, and structure the raw data gathered by the crawler

- Identify and extract the meaningful pieces of information from preprocessed text.

- Integrate the extracted facts into a structured, machine-processable knowledge graph.

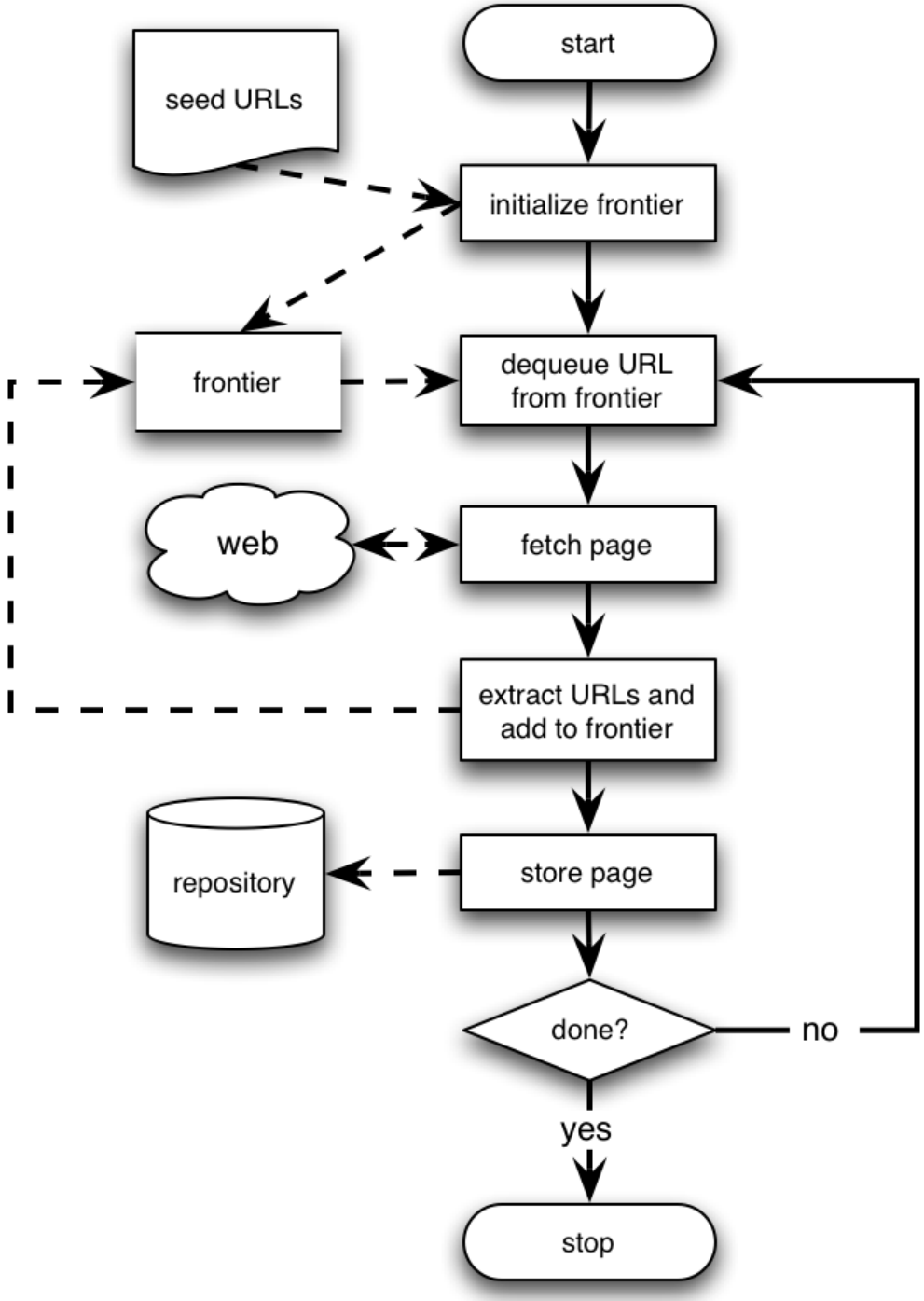
ESILV

Web crawling/ scraping

ESILV

How Web Crawlers Work

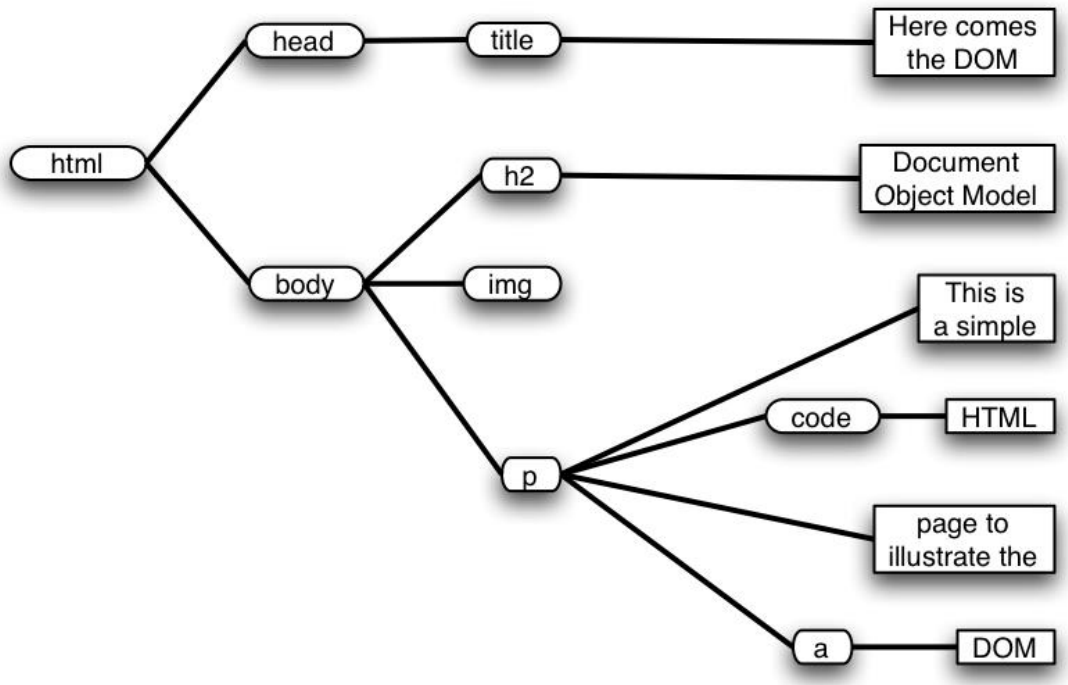
- This is a sequential crawler
- Seeds can be any list of starting URLs
- Fetching Pages: The crawler downloads web pages by sending HTTP requests.
- Order of page visits is determined by frontier data structure
- Stop criterion can be anything



How Web Crawlers Work

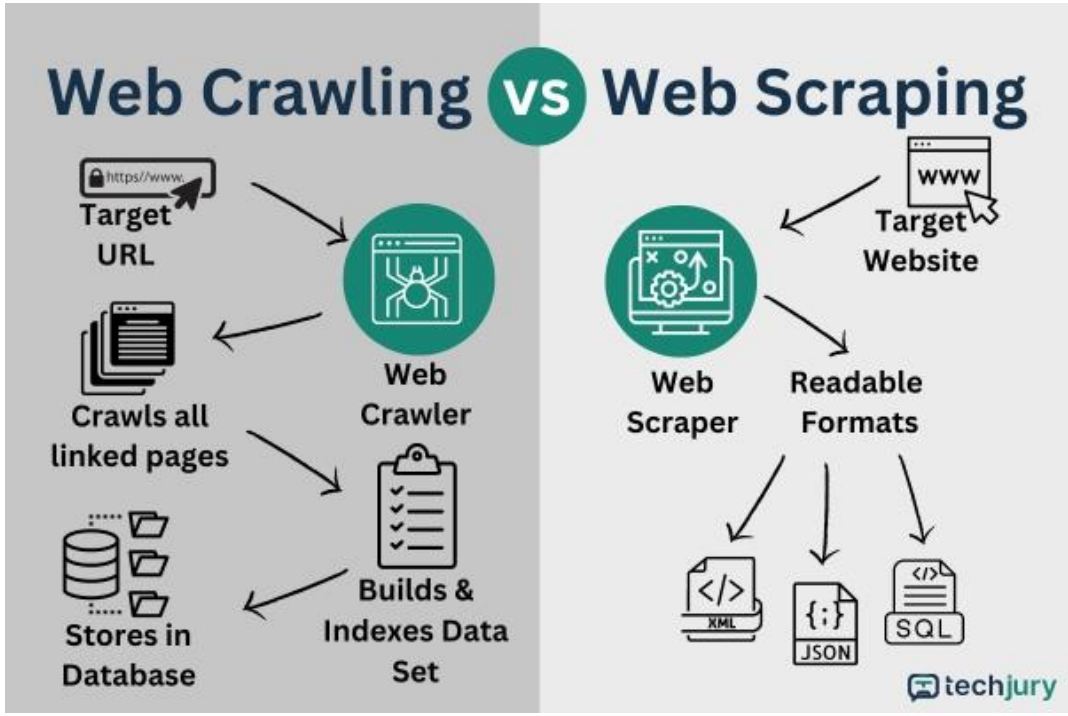
- Parsing & Link Extraction:
  - Use HTML parsers (e.g., **BeautifulSoup**) to extract all the links (anchor tags) from the page.
  - Filter out irrelevant or duplicate URLs.
- Queue Management:
  - Discovered URLs are added to a frontier (queue) to be crawled later.
  - Scheduling algorithms determine the order of crawling.

```
<html>
  <head>
    <title>Here comes the DOM</title>
  </head>
  <body>
    <h2>Document Object Model</h2>
    <img align="right" alt="dom pict" src="dom.png">
    <p>
      This is a simple
      <code>HTML</code>
      page to illustrate the
      <a href="http://www.w3.org/DOM/">DOM</a>
    </p>
  </body>
</html>
```



Web scraping

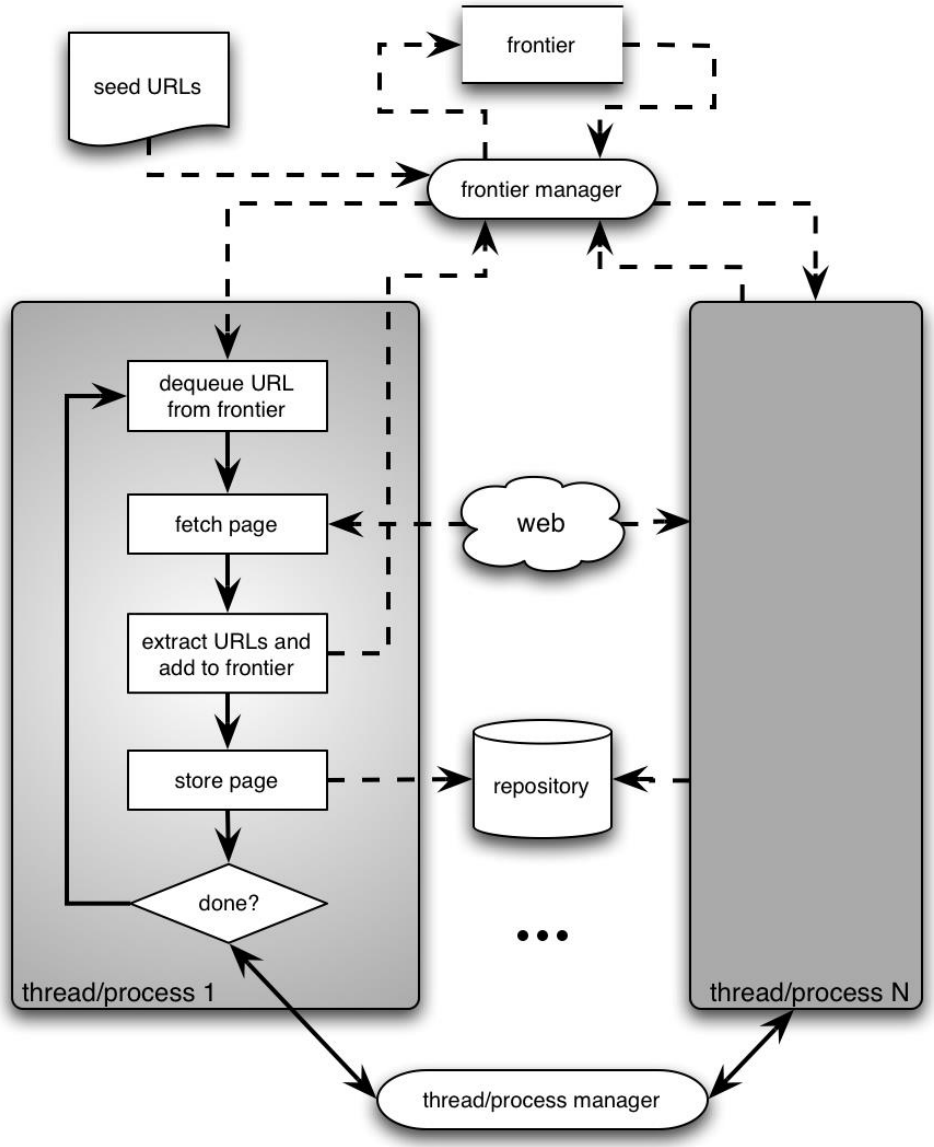
- Web scraping is the process of using bots to extract **specific data** from **a target website**. Web scraping uses web crawlers to scan and store all the content from a webpage (or a set of webpages) and then uses a web scraping tool or script to extract specific data from that content.



ESILV

Advanced Crawling & Data Collection

- **Handling Dynamic Content:** Modern websites use AJAX/JavaScript
  - **Solution:** Employ headless browsers (e.g., Puppeteer, Selenium) to render pages
- **Distributed Crawling Architectures:** Utilize multiple agents to cover the web efficiently
  - Address challenges such as load balancing, duplicate removal, and error recovery
- **Dealing with Duplicates:**
  - Use content hashing (e.g., MD5, SHA-1) to detect and remove duplicate pages.



More issues on crawling & data collection

- Politeness, Ethical & Legal Considerations:
  - Crawlers must respect the guidelines specified in a website's **robots.txt** file.
  - Implement delays between requests to avoid overloading a server. (you a not conducting a DDoS attack!)
  - Avoid collecting personal data without permission, and ensure that the crawling activity complies with legal requirements.

A robots.txt file tells search engine crawlers which URLs the crawler can access on your site.
Here for more info on robots.txt

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /junk/
```

- The best way of learning web crawling is implementing by yourselves.

ESILV

Some Python packages for web crawling/scraping

- *Requests*: making http request
- BeautifulSoup: parsing the html content
- Selenium: automating web browser (with user defined header)
- Lxml module: Paring XML and HTML with high performance. (an XML used for storage can be very large)
- Urllib: working with URLs
- PyautoGUI: Controlling the mouse and keyboard to automate tasks
- Schedule: scheduling python functions to run for periodical or interval missions
- More on https://www.geeksforgeeks.org/python-web-scraping-tutorial/

ESILV

Tools and frameworks

- Scrapy: An open-source framework in Python for large-scale crawling and web scraping.
- Apache Nutch: A highly extensible and scalable open-source web crawler built on **Hadoop**. (In Java)
- Heritrix: The Internet Archive’s crawler, designed for archival purposes.

Data (text) pre-processing

ESILV

Data (text) pre-processing: From raw to structured text

1. Word (term) extraction/segmentation
   - easy for space-delimited languages (*eg.* English and French)
   - Not straightforward for *scriptio continua* languages (*eg.* Chinese)
     - *Using probability models (eg. HMM+Viterbi) or sequential models (eg. LSTM, Transformer, etc.)*
2. Stop-words removal
3. Stemming/lemmatization
4. Word representation:
   - Tokenization
   - Frequency counts and computing TF-IDF term weights.

ESILV

Stop-words removal

- Many of the most frequently used words in English are useless in IR and text mining – these words are called stop-words.
  - the, of, and, to, ….
  - Typically, about 400 to 500 such words
  - For an application, an additional domain specific stopwords list may be constructed
- Why do we need to remove stop-words?
  - Reduce indexing (or data) file size
    - Stop-words accounts 20-30% of total word counts.
  - Improve efficiency and effectiveness
    - Stop-words are not useful for searching or text mining
    - they may also confuse the retrieval system.

ESILV

Stemming

- Techniques used to find out the root/stem of a word. E.g.,
  - user — engineering
  - users — engineered
  - used — engineer
  - using
- stem: use — engineer

Usefulness:

- improving effectiveness of IR and text mining
  - matching similar words
  - Mainly improve recall
- reducing indexing size
  - combing words with same roots may reduce indexing size as much as 40-50%.

Basic stemming methods

Using a set of rules. E.g.,

- remove ending
  - if a word ends with a consonant other than s, followed by an s, then delete s.
  - if a word ends in es, drop the s.
  - if a word ends in ing, delete the ing unless the remaining word consists only of one letter or of th.
  - If a word ends with ed, preceded by a consonant, delete the ed unless this leaves only a single letter.
  - …...
- transform words
  - if a word ends with “ies” but not “eies” or “aies” then “ies --> y.”

ESILV

Lemmatization (1)

- Grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma
  - Similar to Stemming, but more precise, with the consideration of the context
    - "better" has "good" as its lemma. This link is missed by stemming, as it requires a dictionary look-up.
    - "walk" is the base form for the word "walking", this is matched in both stemming and lemmatization.
    - The word "meeting" can be either the base form of a **noun** or a form of a **verb** ("to meet") depending on the context.
- Algorithms
  - A trivial way: dictionary lookup
  - Rule-based system (based on **PoS tagging**)

ESILV

Lemmatization (2): Algo. by PoS tagging

- Marking up a word in a text (corpus) as corresponding to a particular **P**art **o**f **S**peech, based on both its definition and its context.
- Such tagging helps finding the lemma of words
- Example, with more tags here
- How to find the PoS tagging? We will see later in the named entities recognition part

Tag	Description	Example
NN	Singular common noun	dog
PRP	Peronal Pronoun	He, she, it
VB	Base form of a verb	Run
VBD	Past tense verb	ran
JJ	Adjective	quick
RB	Adverb	quickly
DT	Determiner (eg., articles)	the, a
IN	Preposition	in, on, because
UH	Interjection	Oh, wow

ESILV

Example

- Original Text: The quick brown foxes are jumping over the lazy dogs.
  Lemmatized Text: the quick brown fox be jump over the lazy dog .

The → Determiner (DT)
quick → Adjective (JJ)
brown → Adjective (JJ)
fox → Noun (NN)
jumps → Verb (VBZ)
over → Preposition (IN)
the → Determiner (DT)
lazy → Adjective (JJ)
dog → Noun (NN)

ESILV

Lemmatization (3): implementation with WordNet Lemmatizer

- The Natural Language Toolkit (**NLTK**) and **spaCy** provides a class called `WordNetLemmatizer`, which is a thin wrapper around the WordNet corpus, based on *dictionary lookup*

```
# import these modules
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos="a"))
```

```
rocks : rock
corpora : corpus
better : good
```

pos*(str)*-the PoS tag*:*
"n" for nouns,
"v" for verbs,
"a" for adjectives,
"r" for adverbs and
"s" for satellite adjectives.

ESILV

Frequency counts + TF-IDF

- Counts the number of times a word occurred in a document.
  - Using occurrence frequencies to indicate relative importance of a word in a document.
    - if a word appears often in a document, the document likely “deals with” subjects related to the word.
- Counts the number of documents in the collection that contains each word
- TF-IDF can be computed.
  - (Term Frequency-Inverse Document Frequency. It quantifies a word’s importance in a document by balancing its frequency in that document with its rarity across the entire collection.)
  - Brief example in the next slide

ESILV

TF-IDF

Document 1:
"Neural networks are a key component of deep learning. Neural models can learn complex patterns."
Document 2:
"Football is a popular sport. Fans enjoy exciting matches and competitive games."

Step1: tokenization and count	Frequency of "neural" in Document 1: 2 occurrences Total tokens in Document 1: 15	Frequency of "neural" in Document 2: 0 occurrences Total tokens in Document 2: 12
Step 2: calculate TF	TF for "neural" in Document 1: TFD1("neural")=2/15≈0.1333	TF for "neural" in Document 2: TFD2("neural")=0/12=0
Step 3: Calculate IDF	Number of documents (N): 2 Document frequency for "neural" (df): It appears in Document 1 only, so df(“neural”)=1. IDF(“neural”)=ln(N/df)=ln(2/1) ≈0.693	
Step 4:Calculate TF-IDF	$TF\text{-}IDF_{D1}$("neural") $=TF_{D1}\times IDF\approx 0.1333\times 0.693\approx 0.0924$	$TF\text{-}IDF_{D2}$("neural")=0×0.693=0

- Application: Find tokens with top-k TF-IDF to represent a document

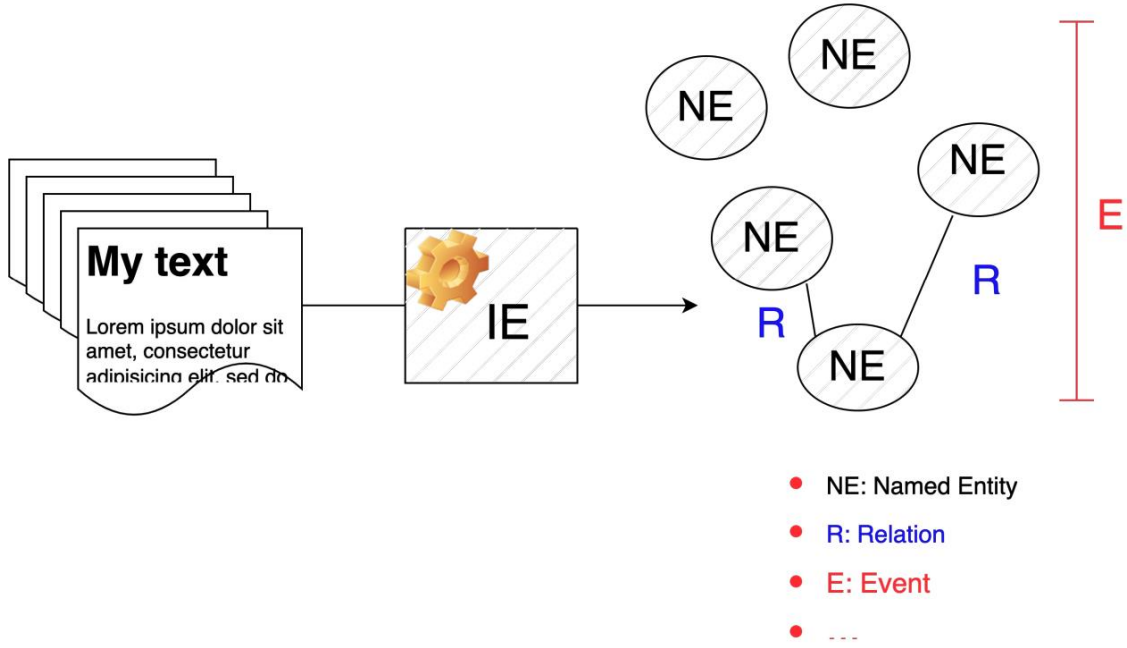
ESILV

Information Retrieval

ESILV

Information Extraction (IE) in practice: NE

- **Named entity**: A lexical unit which refers to a real-word object classified within a category: people, organization, location, money, percent, etc
- **Named Entity Recognition (NER)**: a salient sub-task of IE. Involves three steps: recognition, categorization, normalization (if needed)



"Epidemie Ebola : 2013-2015

Cette épidémie, commencée en décembre 2013, est parfois qualifiée d'atypique », parce que non maîtrisée. En juillet 2014, elle évolue de manière préoccupante en Guinée, au Liberia et en Sierra Leone. Le 20 août 2014, 844 décès étaient officiellement confirmés comme dus au virus69. Une flambée se déclare dans le district de Boende."

Disease
Location
Date

ESILV

Information Extraction (IE) in practice: Relation

- Relations:
  - Facts generally involving named (named) entities
  - Facts as n-ary relations between entities
  - Literature focuses on binary relations, converted into knowledge triples:

    `(subject, predicate, object)`

Plusieurs manifestants sont désignés comme porte-parole par des Gilets jaunes ou sont mis en avant par les médias. Au début du mouvement, les principales personnalités médiatisées "sont Éric Drouet, qui a lancé sur Facebook l'appel au rassemblement du 17 novembre 2018, Priscillia Ludosky, à l'origine de la pétition en ligne de mai 2018 appelant à la baisse des prix du carburant, Jacline Mouraud, dont la vidéo à l'adresse d'Emmanuel Macron est devenue virale

subject	predicate	object
Eric Drouet	personnalité	Gilets Jaunes
Priscillia Ludosky	personnalité	Gilets Jaunes
Jacline Mouraud	personnalité	Gilets Jaunes
Eric Drouet	Origine	Rassemblement
Priscillia Ludosky	Origine	Petition

ESILV

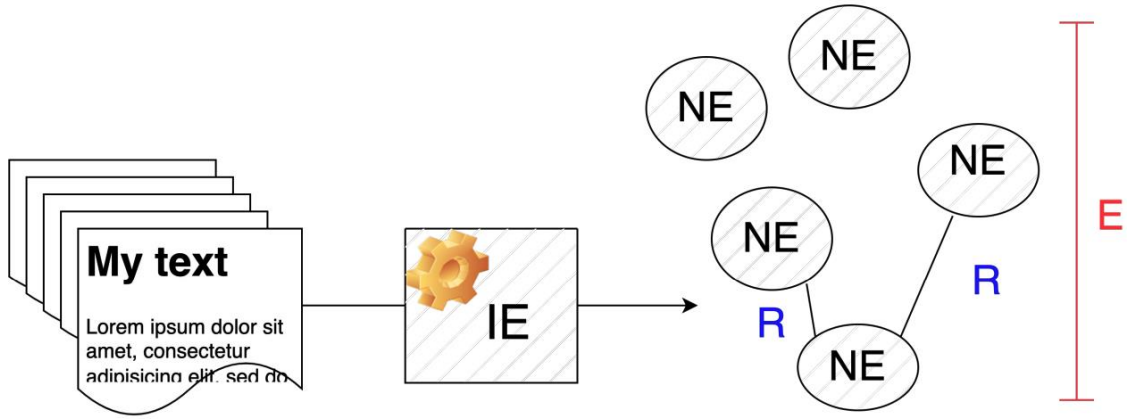
Information Extraction (IE) in practice: Events

- There is no standard definition for Event, it is a specific occurrence triggered at a place and time
- An award event ”Cedric Viliani from France, won the Fields Medal in 2010”
  - Entities: objects involved (Cedric Villiani, Fields Medal)
  - Event type: nature of the event (wining)
  - Event trigger: a verb or noun as the core unit of the evnt (won)
  - Event arguments: attributes of the event.(2010)
  - Argument role: relationship between the event and the argument eg. time, date, place, (2010: date)

ESILV

Extracting Named Entities: NE recognition (NER)

- The task of NER



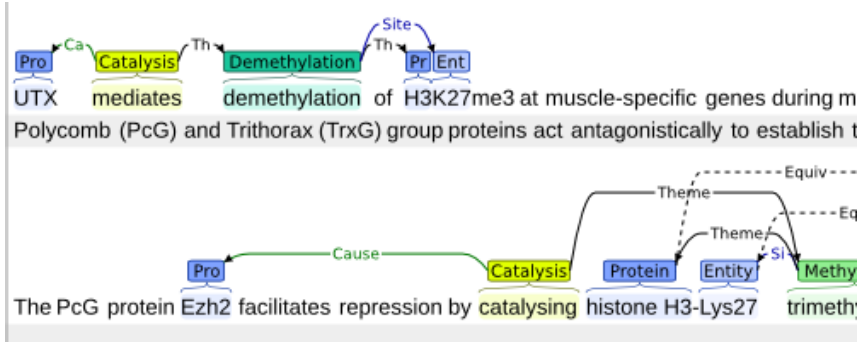
- NE: Named Entity
- R: Relation
- E: Event
- ---

- How to do NER?

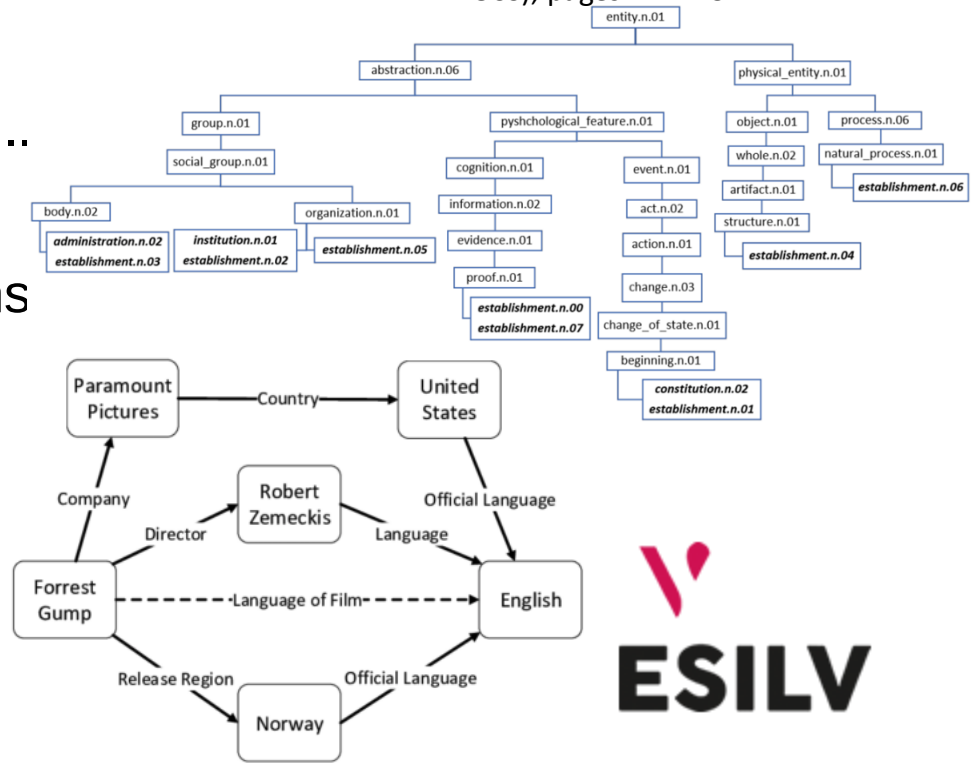
ESILV

What are data resources like (as supervision)?

- Annotated documents
  - Human-annotated documents
- Lexical bases
  - Thesaurus: words and semantic relations
  - Terminologies, lexical databases: (domain-)related words, **entities** and **relations** (eg. UMLS, WordNet, ...
- Knowledge graphs
  - Concepts, real-world entities, events and rich relations
  - Knowledge of the web (Freebase, DpBedia,Yago, ...)
  - Domain-knowledge



© BRAT tool, https://brat.nlplab.org/examples.html



© WordNet, Ekmekci et Howald, Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS), pages 147–157

ESILV

Approaches for NER

- **Rule-based**
  - Manual/hand-written patterns
- Unsupervised learning
  - Mostly clustering techniques
  - Discriminative models
- Feature-based supervised learning
  - Naive Bayes, Hidden Marko Models (HMM), Conditional Random Field (CRF), Maximum-Entropy Markov Models (MEMMs)
- (Deep) Neural - based learning
  - Distributed representation of inputs, deep encoding of context, tags

Approaches for NER: Rule-based methods

- A rule describes a pattern for a NE category (Org, Per, etc.)
  - Recognizes the applicability of the rule:
    - Matches the instances of the category based on regularities
  - If the context is applicable then:
    - Bound the NE: <Begin /NE….End/NE>
    - Assign the NE to the category
- Features used for NER
  - Lexical features
    - Position, punctuation, spelling, ...
  - Syntaxic features
    - Prefix, suffix, grammatical tag, negation, ..
  - Semantic features
    - Polarity (positive, negative, neutral), relation with word contexts, ...

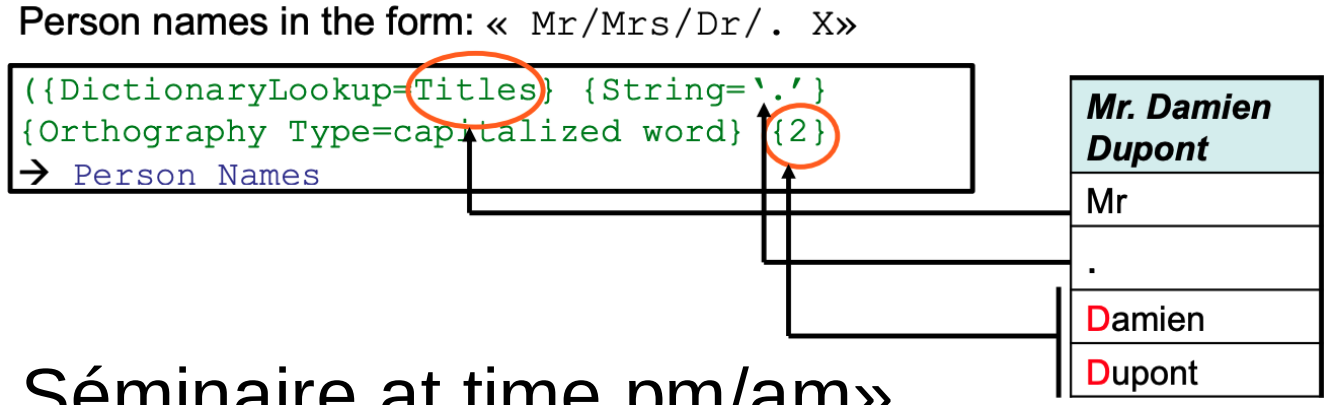
Example

Date	
Year	`(aaaa)`
Month	`(m-nom dans {Janvier…Décembre})`
Month-no	`(m-number=integer and 0<m-number<=12)`
Day	`(j-name dans {Lundi…Dimanche})`
Day-no	`(j-number=integer and 1<m-number<=long(mois));`

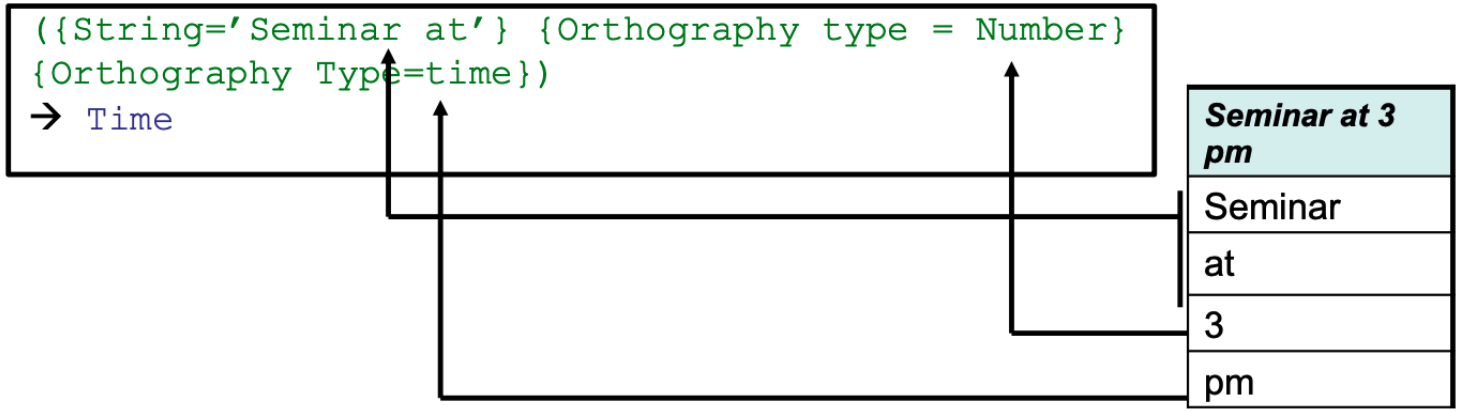
ESILV

Rule-based methods: Examples

- Person names in the form: « Mr/Mrs/Dr/. X»



- Seminar schedule: « Séminaire at time pm/am»



Rule-based methods: Examples

- Some NE requires a pair o rules: `Begin` and `End` for tagging and bounding
  - Expression: « To appear ‘Le monde ’ Vol. 5 »
- Some patterns allow bounding a set of NE (eg. rich tabular forms)
  - Recognize room number and renting price

```
({String=’to appear in’})jstart ({Orthographe
Type=Capitalized word} {2-5})
→ Insert <journal> after:jstart
```

```
({tag=<journal>}) jstart ({Orthographe
Type=word}+):jend {String=‘vol’})
→ Insert </journal> after:jend
```

```
({Ortography
Type=Digit}):Bedrooms({String=‘BR’})({}*)({String=‘$’
})({Ortography Type=Number}):Price
→ Number of Bedrooms=:Bedroom, Rent=:Price <journal>
```


ESILV

Rule-based methods: implementation

- Extract by Regular Expression (RE): More details
- Example:
  - Extract organization names

```
In [1]: import re

In [2]: example = "Department of Computer Science. Faculty of Science and En
   ...: gineering, ESILV, France"

In [3]: org = re.search(r"([AZ][^\s,.]+[.]?\s[(]?)*(Dept|Association|Departm
   ...: ent|University|Faculty)[^,\d]*?(?=,|\.|\d)", example).group(0)

In [4]: print(org)
Department of Computer Science
```

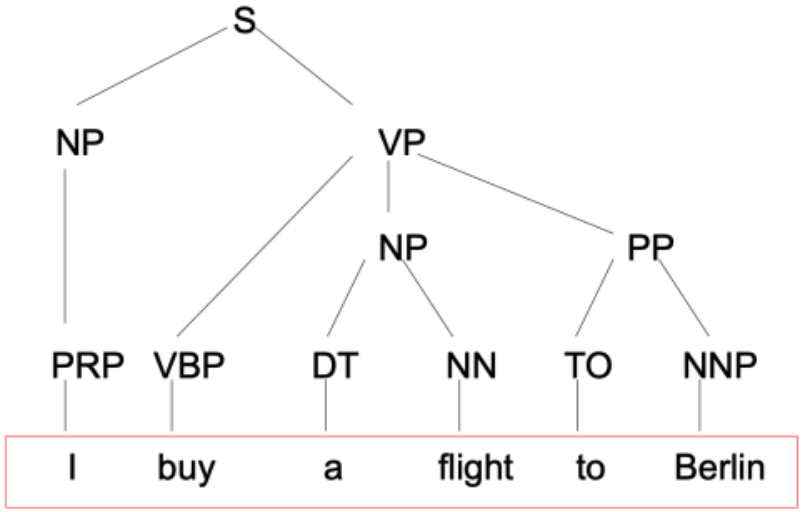
  - Extract e-mail addresses

```
In [12]: str='purple alice@google.com, blah monkey bob@abc.com blah dishwasher

In [13]: emails = re.findall(r'[\w\.-]+@[\w\.-]+', str)

In [14]: print(emails)
['alice@google.com', 'bob@abc.com']
```


ESILV

Rule-based methods: Using NLP tags

- NLP might help designing hand-written rules
  - Uses NLP tags
    - Part Of Speech (POS) tagging
    - Syntactic parsing: mark words as nouns, verbs, preposition, etc.
    - Semantic word categories from an external resource (Yago, Freebase, WordNet, etc.)
  - Rule-based expressions with NLP
    - Which person holds what office in what organization: `[Pers], [office] of [Org]`
    - Where an organization is located: `[Org] in [Loc]`



ESILV

Limitations of rule-based methods

- Historically (1987-1997) from Message Understanding events (MUC). Main tasks focused on extracting from news:
  - NER, Coreference resolution, Information Extraction
- Limitations of rule-based methods
  - Cost of the definition of hand-written rules: 1 rule per form ! Useful for limited "clear" patterns
  - Recall might be low in rich texts
  - Gradually moved to machine-learning based methods

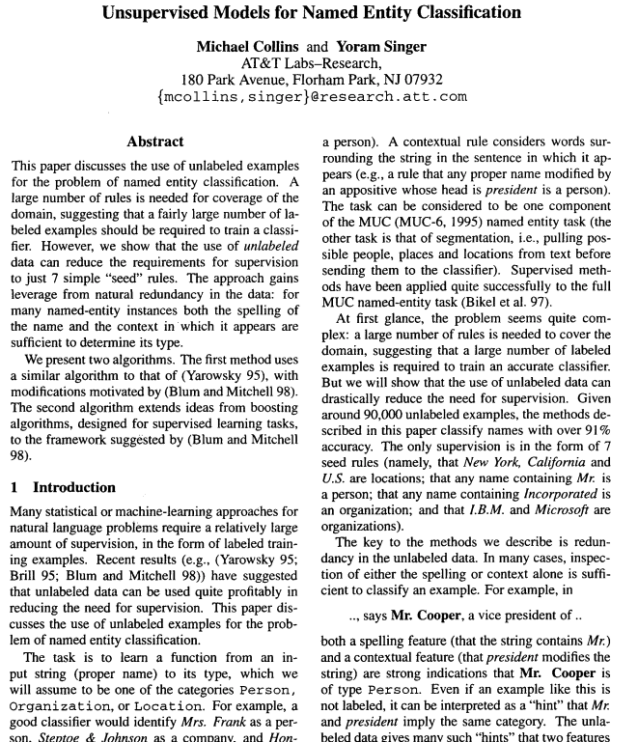
ESILV

Approaches for NER

- Rule-based
  - Manual/hand-written patterns
- **Unsupervised learning**
  - Mostly clustering techniques
  - Discriminative models
- Feature-based supervised learning
  - Naive Bayes, Hidden Marko Models (HMM), Conditional Random Field (CRF), Maximum-Entropy Markov Models (MEMMs)
- (Deep) Neural - based learning
  - Distributed representation of inputs, deep encoding of context, tags

NER-Unsupervised learning methods

- Key idea
  - No human-labeled text
- General pipeline
  - Select features to encode/represent input text
    - Manual/hand-written rules
  - Apply a clustering model to feature-based descriptors
    - The model uses a combination of features
    - The model assigns to tokens in the input text **automatically-generated labels**
- Generally, learns one set of rules per NE category



Unsupervised Models for Named Entity Classification

Michael Collins and **Yoram Singer**
AT&T Labs–Research,
180 Park Avenue, Florham Park, NJ 07932
{mcollins,singer}@research.att.com

Abstract

This paper discusses the use of unlabeled examples for the problem of named entity classification. A large number of rules is needed for coverage of the domain, suggesting that a fairly large number of labeled examples should be required to train a classifier. However, we show that the use of *unlabeled* data can reduce the requirements for supervision to just 7 simple "seed" rules. The approach gains leverage from natural redundancy in the data: for many named-entity instances both the spelling of the name and the context in which it appears are sufficient to determine its type.

We present two algorithms. The first method uses a similar algorithm to that of (Yarowsky 95), with modifications motivated by (Blum and Mitchell 98). The second algorithm extends ideas from boosting algorithms, designed for supervised learning tasks, to the framework suggested by (Blum and Mitchell 98).

1 Introduction

Many statistical or machine-learning approaches for natural language problems require a relatively large amount of supervision, in the form of labeled training examples. Recent results (e.g., (Yarowsky 95; Brill 95; Blum and Mitchell 98)) have suggested that unlabeled data can be used quite profitably in reducing the need for supervision. This paper discusses the use of unlabeled examples for the problem of named entity classification.

The task is to learn a function from an input string (proper name) to its type, which we will assume to be one of the categories `Person`, `Organization`, or `Location`. For example, a good classifier would identify *Mrs. Frank* as a person, *Steptoe & Johnson* as a company, and *Hon-*

a person). A contextual rule considers words surrounding the string in the sentence in which it appears (e.g., a rule that any proper name modified by an appositive whose head is *president* is a person). The task can be considered to be one component of the MUC (MUC-6, 1995) named entity task (the other task is that of segmentation, i.e., pulling possible people, places and locations from text before sending them to the classifier). Supervised methods have been applied quite successfully to the full MUC named-entity task (Bikel et al. 97).

At first glance, the problem seems quite complex: a large number of rules is needed to cover the domain, suggesting that a large number of labeled examples is required to train an accurate classifier. But we will show that the use of unlabeled data can drastically reduce the need for supervision. Given around 90,000 unlabeled examples, the methods described in this paper classify names with over 91% accuracy. The only supervision is in the form of 7 seed rules (namely, that *New York*, *California* and *U.S.* are locations; that any name containing *Mr.* is a person; that any name containing *Incorporated* is an organization; and that *I.B.M.* and *Microsoft* are organizations).

The key to the methods we describe is redundancy in the unlabeled data. In many cases, inspection of either the spelling or context alone is sufficient to classify an example. For example, in

.., says **Mr. Cooper**, a vice president of ..

both a spelling feature (that the string contains *Mr.*) and a contextual feature (that *president* modifies the string) are strong indications that **Mr. Cooper** is of type `Person`. Even if an example like this is not labeled, it can be interpreted as a "hint" that *Mr.* and *president* imply the same category. The unlabeled data gives many such "hints" that two features

M. Collins, Y. Singer; Unsupervised models for named entity classification EMNLP 1999

ESILV

The Collins & Yoram method

- Key idea
  - High-level of redundancy in the data makes possible to automatically generate/learn "labelling rules" ie. assign a NE category (`Person, Location, Organization`) to a token à well known today as self-supervision
- Key notion: feature-based rule
  - Parse sentences/strings in the text: recognize capitals, grammatical positions (Noun, Verb, Preposition, modifiers, etc.
  - Feature-based predicate: indicates the properties of the string on a pair of **(spelling, context)** pair
    - Spelling: forms of the words within the string
    - Context: forms and roles of the words surrounding the sentence
  - Feature-based rule: feature-based predicate that automatically assigns to the string/sentence **an entity category**

ESILV

Sentence	Entities (Spelling/Context)	Features
But Robert Jordan, a partner at Steptoe & Johnson who took ...	Robert Jordan/partner	full-string=Robert_Jordan contains(Robert) contains(Jordan) context=partner context-type=appos
	Steptoe & Johnson/partner_at	full-string=Steptoe_&_Johnson contains(Steptoe) contains(&) contains(Johnson) nonalpha=& context=partner_at context-type=prep
By hiring a company like A.T.&T. ...	A.T.&T./company_like	full-string=A.T.&T. allcap2 nonalpha=..&. context=company_like context-type=prep
Hanson acquired Kidde Incorporated, parent of Kidde Credit, for ...	Kidde Incorporated/parent	full-string=Kidde_Incorporated contains(Kidde) contains(Incorporated) context=parent context-type=appos
	Kidde-Credit/parent_of	full-string=Kidde_Credit contains(Kidde) contains(Credit) context=parent_of context-type=prep

Feature-based descriptors of strings

```
full-string=New_York      →   Location
full-string=California    →   Location
full-string=U.S.          →   Location
contains(Mr.)             →   Person
contains(Incorporated)    →   Organization
full-string=Microsoft     →   Organization
full-string=I.B.M.        →   Organization
```

Feature-based EN labelling rules

ESILV

The general algorithm of automatic generation of labeling rules

– Input: training **T** text, **R** a set of seed rules; **n**: number of rules generated at each iteration, **k**: number of labels

Begin

1- Automatically label the training texts **T** using **R**

2- Induce contextualized rules from labeled data

3- Select the most confident contextual rules (less than **n***k**)

4- Apply the contextual rules generated in 3 to automatically label text **T**

5- Select the most confident spelling rules (less than **n*k**)

6- n=n+5 ; if n<threshold

Add the n top selected spelling rules to R/spelling and contextual rules;

Return to step 1.

End

ESILV

- Advantages
  - Limited amount supervision/labels is initially needed
  - Possibility of using a wide range of supervised learning algorithms for self-generating new labeling rules (naive bayes, decision trees, bootstrapping)
- Limitations
  - Performance heavily depends on seed rules
  - Lack of generalizability, exhaustivity of rules: diversity of contexts lead to noisy rules
  - Requires the definition of seed hand-written rules ...inherently induces limitations of this method!

ESILV

Approaches for NER

- Rule-based
  - Manual/hand-written patterns
- Unsupervised learning
  - Mostly clustering techniques
  - Discriminative models
- **Feature-based supervised learning**
  - Naive Bayes, Hidden Marko Models (HMM), Conditional Random Field (CRF), Maximum-Entropy Markov Models (MEMMs)
- (Deep) Neural - based learning
  - Distributed representation of inputs, deep encoding of context, tags

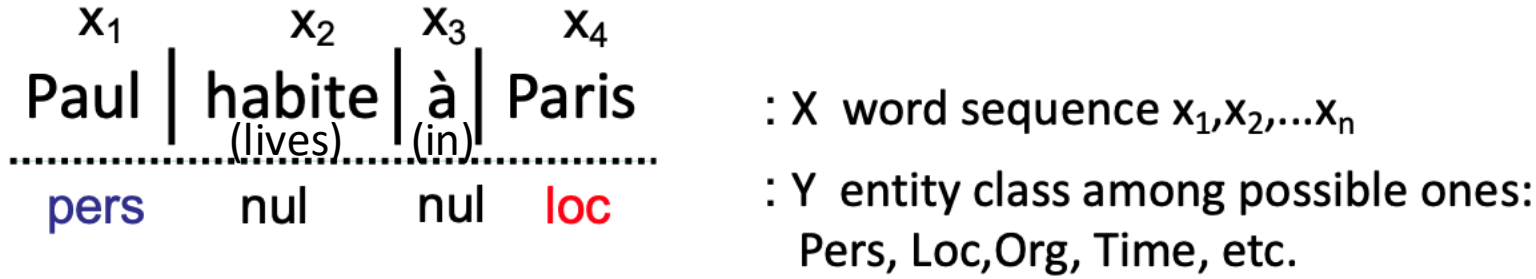
NER-Supervised learning methods

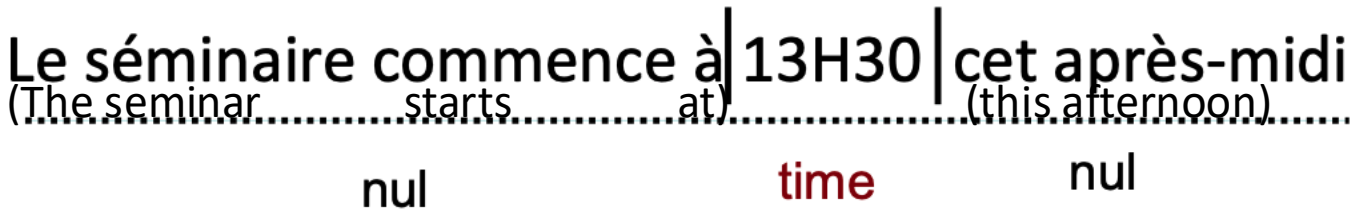
- Key idea
  - Automatically learn features-based models for classifying entities
  - A class is a category label
- General pipeline
  - Identify a set of target NE categories to classify
  - Design relevant features to encode/represent input text (either training or testing) for these NE categories. Build/code the extractors
  - Training
    - Dispose of human-labelled data (training collection): each token is labeled with its category/class
    - Train a text classification model (NB, CRF, HMM, etc.) to predict the entity label class from the data
  - Testing
    - Dispose of a testing collection (to be automatically labeled)
    - Run the trained classification model
    - Output the predicted entities labels/classes

ESILV

Feature Extraction (1)

- Lexical: capitalization, special characters, punctuation, ...
- Syntaxic: POS tags (*e.g.*, V, NN, *etc.*)
- Semantic features: semantic category is a resource,
- Discourse features: computed by using text fragments beyond the sentence.





ESILV

Feature Extraction (2): measure the features

- Build the feature-based descriptors based on feature functions: basic or composite functions, regular expressions, ...
- Features can have Boolean, numerical, range values, ...
- On the current word, on previous words, ? next words (context)

F1 (Y,X,i): feature value x_i with target label Y (trainig data)

Features

```
F1: First letter of xi is capital
F2: xi-1='.'
F3: xi contains special character
F4: length of xi
```

Paul | habite | à | Paris

pers nul nul loc

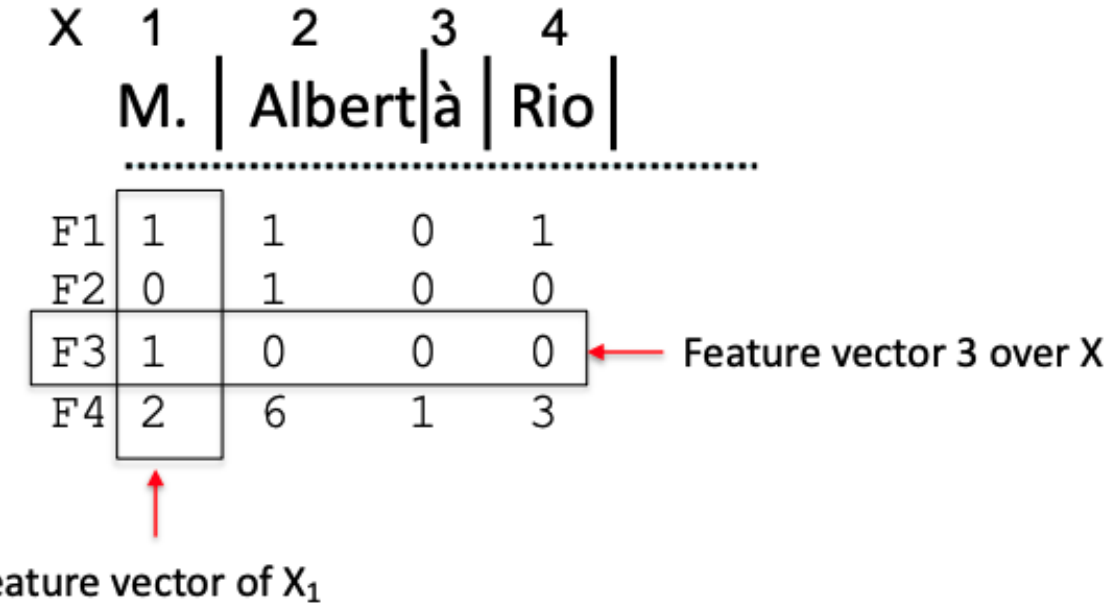
```
F1(pers,X,1)=1   F1(nul,X,2)=0
F2(pers,X,1)=0   F2(nul,X,2)=0
F3(pers,X,1)=0   F3(nul,X,2)=0
F4(pers,X,1)=4   F4(nul,X,2)=6
```

Measure the features

F1 (Y,X,i): feature value x_i with target label Y (trainig data)

Features

```
F1: First letter of xi is capital
F2: xi-1 contains '.'
F3: xi contains special character
F4: length of xi
```



ESILV

Extract features with Python

```
def read_into_feature_dict(file):
    with open(file) as le_file:
        le_dict = {}
        for line in le_file:
            line = line.strip("\n")
            if line not in le_dict:
                # other features besides frequency
                digits = sum(c.isdigit() for c in line)
                spaces = sum(c.isspace() for c in line)
                # initialize an array of [frequency, digits, spaces].
                #Frequency is initially 1
                le_dict[line] = [1,digits, spaces]

            else:
                # increment frequency if we met this before
                le_dict[line][0] = le_dict[line][0] + 1
        return le_dict
```

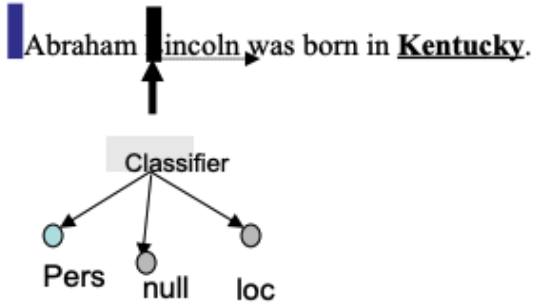
Two features

Dictionary, whose key is the entity, and the value is the feature array

ESILV

Learning methodology: (Very) Naïve Bayes

- Input: n labeled examples in the form (x_i , y_j);
  - x_i : a feature-based descriptor based on m features $\{x_{i1},\ldots,x_{im}\}$;
  - y_j a label in Y=$\{1,2,\ldots,k\}$,
  - k: number of labels.



- Task: Define a function that better maps an input x to output $h(x)$.
  - One way to define $h(x)$ is a conditional model

$$h(x) \backsimeq p(y|x)$$

$$h(x) = argmax_{y \in Y} p(y|x)$$

For a test example x, compute the output as the most likely label

ESILV

Learning methodology: (Very) Naïve Bayes

$$h(x) \backsimeq p(y|x)$$

$$h(x) = argmax_{y \in Y} p(y|x)$$

- How to estimate model parameters? Apply Bayes rule

Prior probability over labels y

Probability of generating input x, given label is y

$$p(x, y) = p(y)p(x|y)$$

independent local models

$$p(x|y) = p(x_1, x_2, \ldots, x_m|y) = \prod_{j=1}^{m} p(x_j|y)$$

$$h(x) = \arg\max_{y \in Y} \tilde{p}(y)\tilde{p}(x|y)$$

- Thus, for a test token x, we can predict the label y, using
  - Where

$$\tilde{p}(y) = \frac{count_y(y)}{\sum count_{y \in Y}(y)}$$

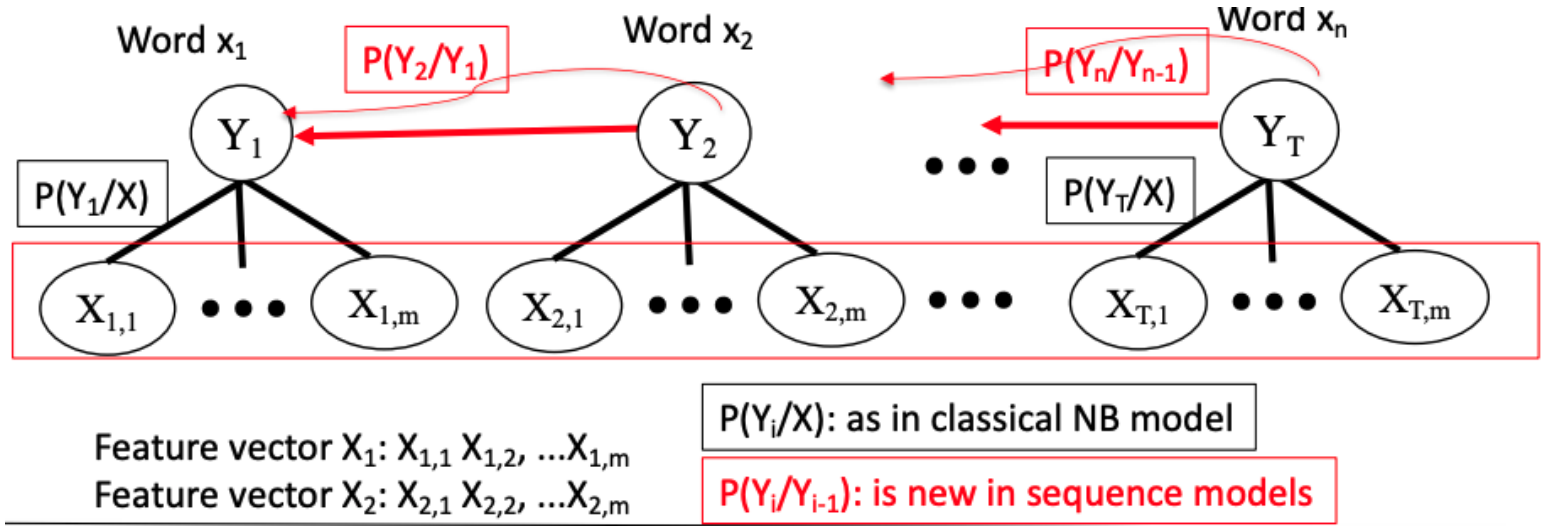
$$\tilde{p}(x|y) = \frac{count(x, y) + \alpha}{\sum_{y \in Y} count(x, y) + \alpha \times m}$$

Smoothed estimation to eliminate issues caused by certain values having 0 occurrences. Here Laplace smoothing for α=1

ESILV

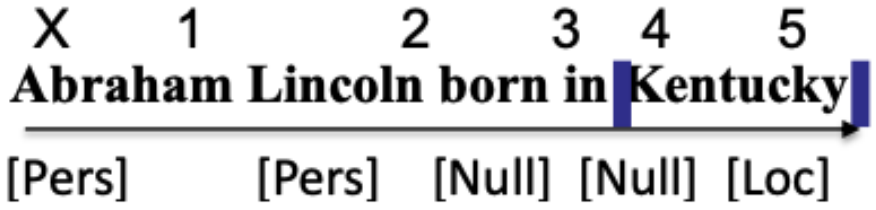
Learning methodology: Conditional Random Fields (CRF)

- In the NB method, texts are not classified as sequences.
- Conditional Random Fields (CRF), applies conditional decision of labelling w.r.t context, i.e., previous entity label
- Input: n labeled sequences in the form (xi , yj);
  - x_i: a feature-based descriptor of word w_i based on m features $\{x_{i1},...,x_{im}\}$;
  - y_i: a label in Y={1,2,...,k},
  - k: number of labels
- Sequence of words/tokens:



Example

Training example

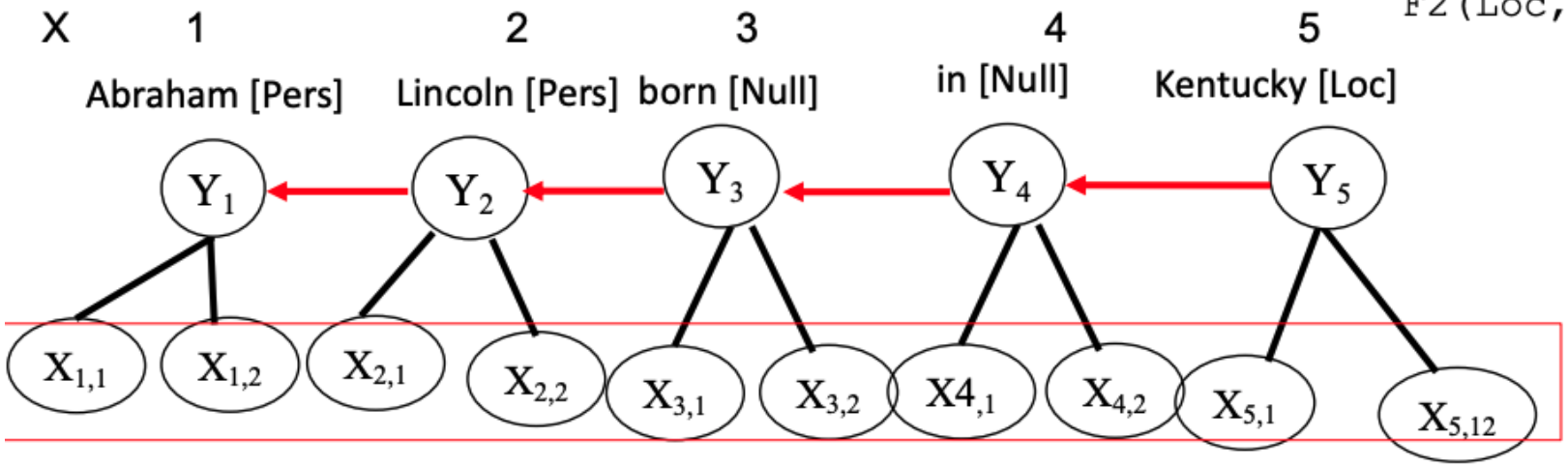


- Sequence in practice
  - x_{1j}: word is capitalised,
  - x_{2j}: word is in lexicon L.

  (Remind: x_{ij}: Feature x_i for target word j)
  - y_1:[Pers], y_2:[Loc], y_3:[Null]

Consider y_i and y_{i-1}

```
F1(pers,X,1)=1          F1(pers,pers,1)=?
F2(pers,X,1)=1          F2(pers,pers,1)=?
F1(pers,X,2)=1          F1(pers,pers,2)=?
F2(pers,X,2)=1          F2(pers,pers,2)=?
F1(null,X,3)=0          F1(null,pers,X,3)=?
F2(null,X,3)=0          F2(null,ers,X,3)=?
F1(null,X,4)=0          F1(null,null,X,4)=?
F2(null,X,4)=0          F2(null,null, X,4)=?
F1(Loc,X,5)=1           F1(Loc,null, X,5)=?
F2(Loc,X,5)=0           F2(Loc,null, X,5)=?
```



ESILV

Learning methodology: CRF (1)

- The features are same as in NB: basic or composite functions, regular expressions…
- For a given input sequence $\boldsymbol{x}=(x_1,x_2,\ldots,x_T)$(e.g., words in a sentence) and a corresponding label sequence $\boldsymbol{y}=(y_1,y_2,\ldots,y_T)$(e.g., PoS tags or entity labels), a linear-chain CRF defines the conditional probability as:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\exp(\sum_{t=1}^{T}\sum_{j=1}^{m}\theta_j F_j(x,y,t))$$

  - It's the value of any feature function for all words of the sentence, indexed by *t*, considering label y
  - Unlike NB model, CRF models impose dependency between target successive labels, with parameters on *t*
  - **Z(x)**: The normalization factor (also called the partition function) which ensures that the probabilities sum to 1:

$$Z(\boldsymbol{x}) = \sum_{y}\exp(\sum_{t=1}^{T}\sum_{j=1}^{m}\theta_j F_j(x,y,t))$$

ESILV

Learning methodology: CRF (2)

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_{t=1}^{T}\sum_{j=1}^{m} \theta_j F_j(x, y, t)\right)$$

- θ_j is the weight associated with the *j*-th feature function, learned during training.
- $F_j(x, y)$ is the scoring function based on the feature function on all words.

$$F_j(x, y) = \sum_{t=1}^{T} f_j(y_t, y_{t-1}, x_t)$$

- $f_j(y_t, y_{t-1}, x_t)$ is the feature extraction function, here are two examples
  - Observation feature: $f_j(y_t, x_t) = 1$ if the word x_t has a certain property and y_t is a specific tag (as in NB method). The transition of the label is not considered.
  - Transition Feature: $f_j(y_t, y_{t-1}, x_t)$=1 if y_{t-1} and y_t follow a certain pattern (e.g., if the previous label is “B-PER” and the current label is ”I-PER”, 0 otherwise). The transition of the label is considered

ESILV

CRF: training

- During training, the weights θ_j are optimized to maximize the likelihood of the training data. The objective function is usually the log-likelihood of the training set:

$$\mathcal{L}(\theta) = \sum_{i}^{N} \log p\left(y^{(i)} \middle| x^{(i)}\right) - \sum_{j=1}^{m} \frac{\theta_m^2}{2\sigma^2}$$

  - N is the number of training examples.
  - The second term is a Gaussian prior (regularization) to prevent over fitting
- Gradient-based methods, such as gradient ascent or quasi-Newton methods, are typically used to optimize the weights.

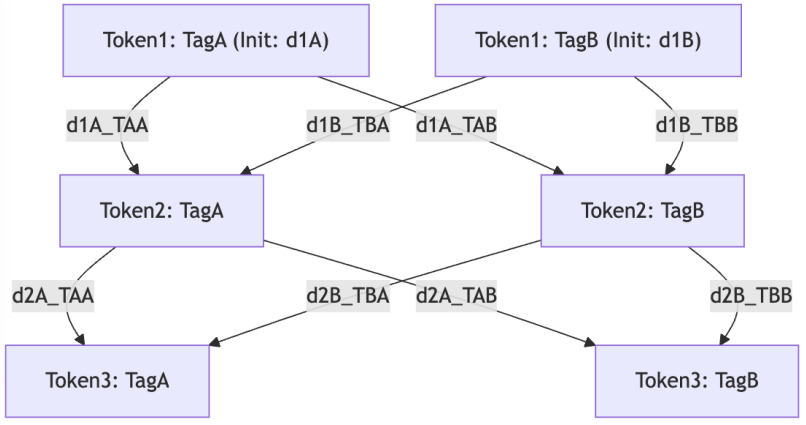
CRF: inference

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|x,\theta) = \frac{\exp(\theta \times F(y,x))}{\sum_{y'} \exp(\theta \times F(y,x))}$$

- We only need to find the label y with the maximum likelihood, so the exponential inside the numerator and the denominator can be ignored

$$\hat{y} = \underset{y}{\operatorname{argmax}} \sum_{t=1}^{T} \theta \times f_j(y_t, y_{t-1}, x_t)$$

- The problem is then to find an entire sequence of labels such that the sum of the transition scores is maximized:
  - use the Viterbi algorithm
- We'll see the same issue in HMM.



Practice: NER pipeline using scikit learn

1. Feature extraction

```
def word2features(sent, i):
    word = sent[i][0]
    postag = sent[i][1]

    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'postag': postag,
        'postag[:2]': postag[:2],
    }
    if i > 0:
        word1 = sent[i-1][0]
        postag1 = sent[i-1][1]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
            '-1:postag': postag1,
            '-1:postag[:2]': postag1[:2],
        })
    else:
        features['BOS'] = True
    if i < len(sent)-1:
        word1 = sent[i+1][0]
        postag1 = sent[i+1][1]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
            '+1:postag': postag1,
            '+1:postag[:2]': postag1[:2],
        })
```

Practice: NER pipeline using scikit learn

- 2. split train and test data

```
X = [sent2features(s) for s in sentences]
y = [sent2labels(s) for s in sentences]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=0)
```

- 3. Train a CRF model

```
crf = sklearn_crfsuite.CRF(
    algorithm='lbfgs',
    c1=0.1,
    c2=0.1,
    max_iterations=100,
    all_possible_transitions=True
)
crf.fit(X_train, y_train)
```

- To see in lab session

ESILV

In summary

- Advantages
  - Naive Bayes methods: naive but effective
  - Experimental results show that CRF have superior accuracy
- Limitations
  - CRFs are much slower to train and do not scale as well to large amounts of training data
- Traditional ML-based classifiers for NER take a set of features as input describing raw text units **U** and outout a prediction of the entity class $\mathbf{C_U}$
- Traditional ML-based classifiers for NER require:
  - Design features to represent raw text units. Combine features for larger units (feeture-engineering)
  - Select a model to combine these features and make a prediction
  - Train the model on a human-labelled corpus
  - Test on unlabeled raw texts

ESILV

Approaches for NER

- Rule-based
  - Manual/hand-written patterns
- Unsupervised learning
  - Mostly clustering techniques
  - Discriminative models
- Feature-based supervised learning
  - Naive Bayes, Hidden Marko Models (HMM), Conditional Random Field (CRF), Maximum-Entropy Markov Models (MEMMs)
- **(Deep) Neural - based learning**
  - Distributed representation of inputs, deep encoding of context, tags

Why deep learning for NER?

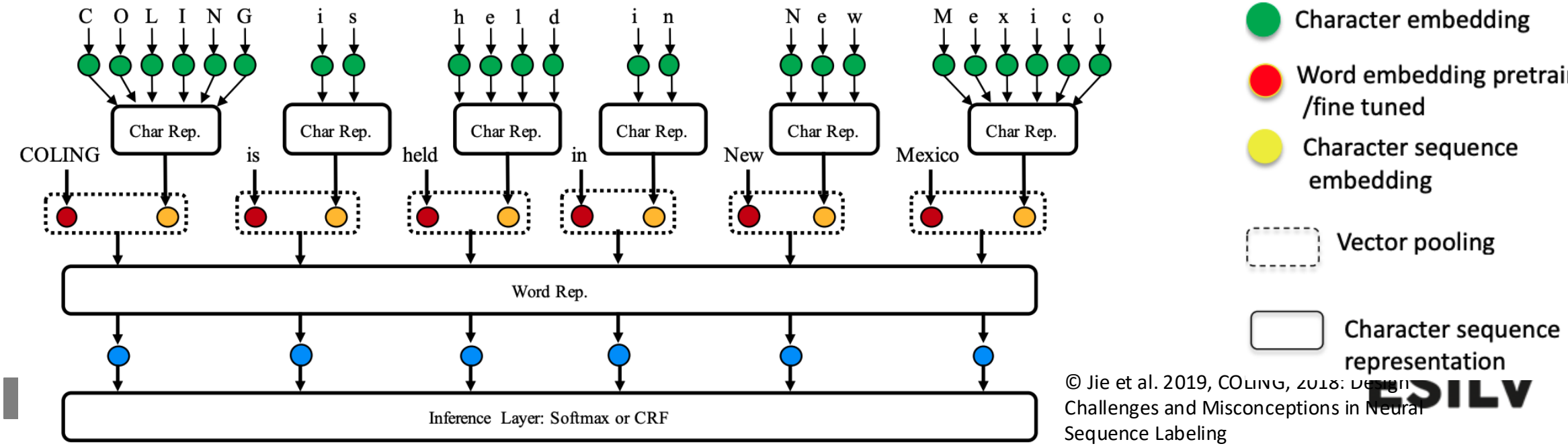
- Learn nonlinear mapping between input text and output labels
  - Naive Bayes, Linear CRF learn linear combinations
- No requirement for feature engineering
  - Is model ineffective or features are not relevant in NB, CRF?
  - DL learns mappings from raw data, not features
- Learning in an end-to-end framework
  - Learn task-based representations of labels: suited representation of texts for NER, RE, ..

- Approaches:
  - Distributed (contextless) representations of input
    - Use **word embeddings** to capture semantic and syntactic properties of words
    - Word2vec, Glove, etc…
  - Sequence-based models of input
    - Consider word in the whole sequence of words (sentence)
    - LSTM, RNN, Pointer architectures mostly
  - Stacked self-attention within context
    - Transformers

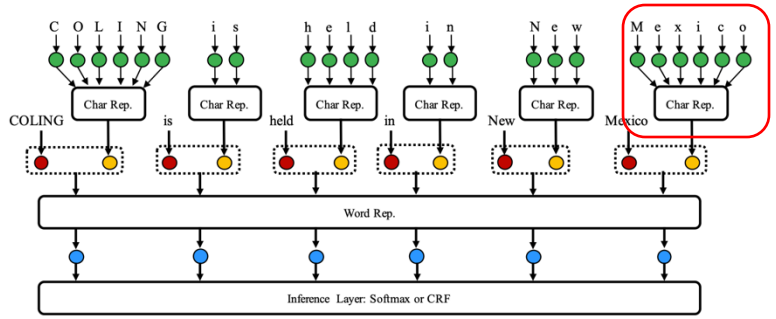
ESILV

Distributed representations of input

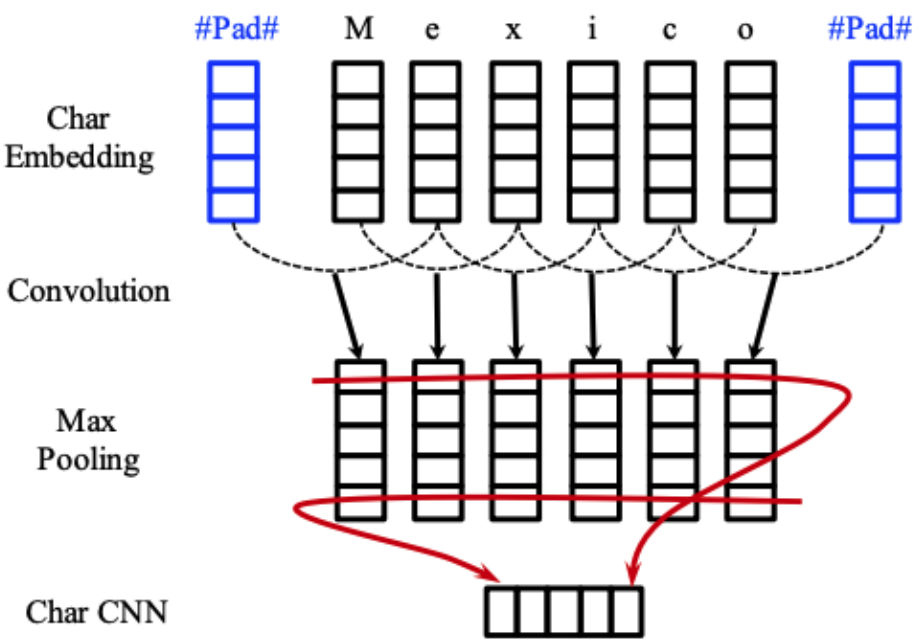
- Use **embeddings** at input (refer to the tutorial)
  - Character embeddings, word embeddings and pool the embeddings
- Output as a classifier prediction
  - Traditional linear combination of inputs (e.g., CRF)
  - Neural inference based on embeddings (e.g., perceptron)



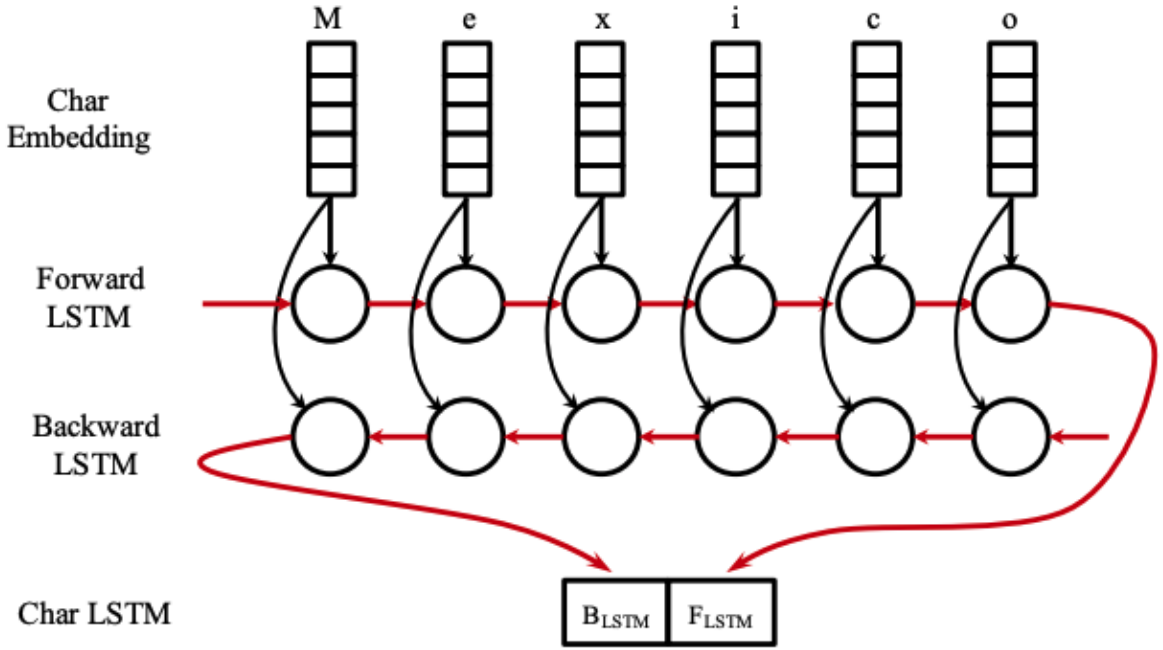
Neural character sequence representations



By using character embedding and word embedding, the model learns Important factor for neural NER either for characters and words representation

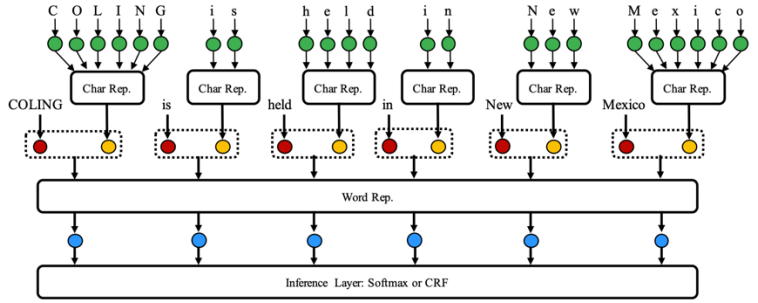


(a) Character CNN.

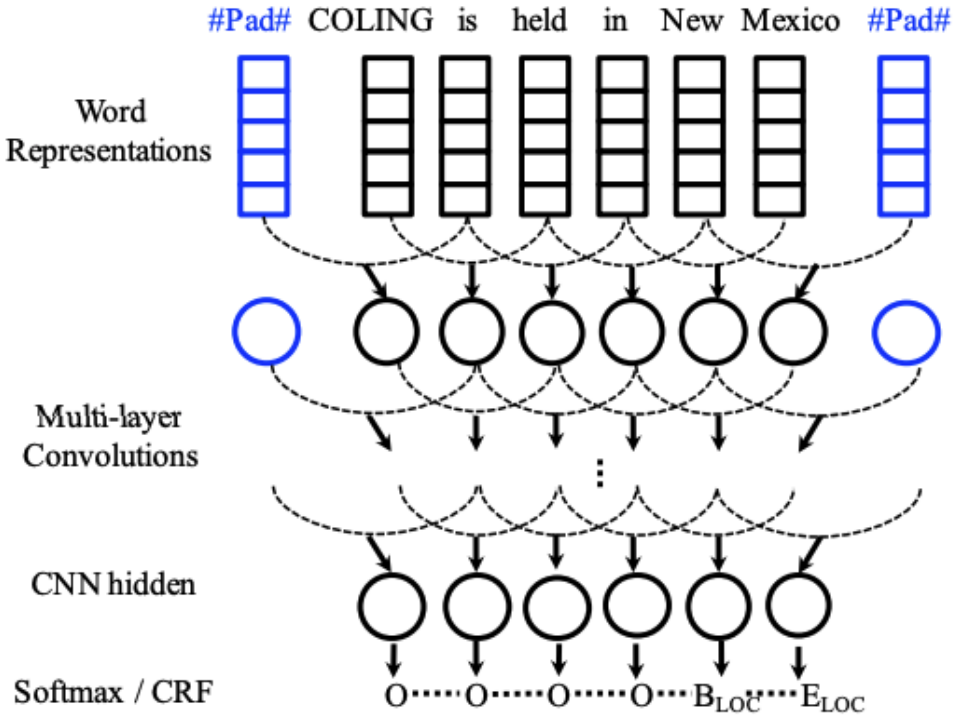


(b) Character LSTM.

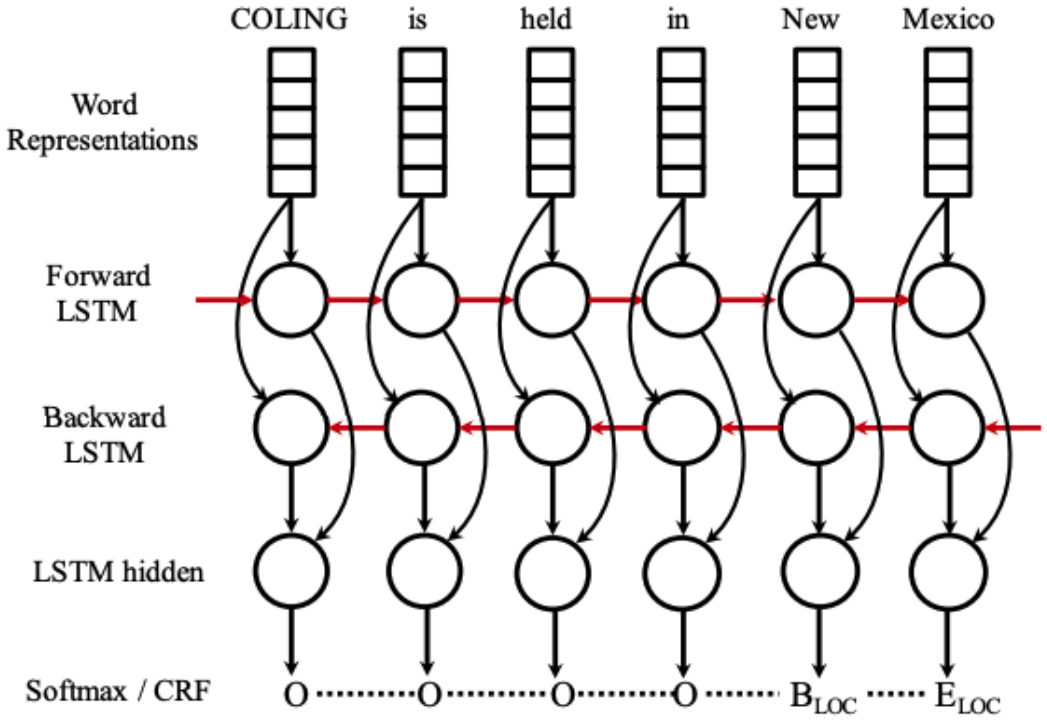
Neural word sequence representations



words are represented by embeddings



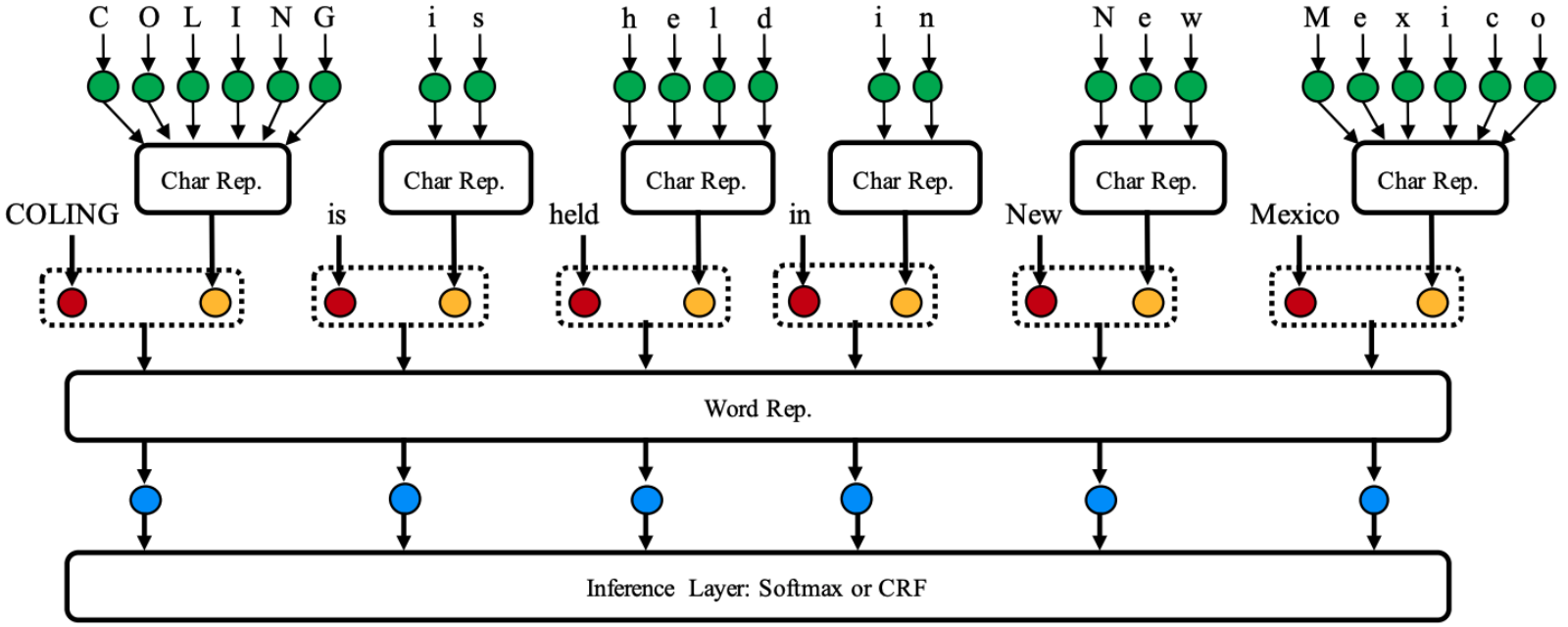
(a) Word CNN.



(b) Word LSTM.

Inference layer

- The inference layer takes the extracted word sequence representations as features and assigns labels to the word sequence.
- Use either softmax or CRF as inference layer



ESILV

Evaluation of link NER

- Measure how much the golden standard annotations map the output of a NER system different scenarios. Build a 2-by-2 contingency matrix.

TP: True Positive FP: False Positive FN: False Negative TN: True Negative		Gold standard	
		Correct	Not correct
NER system	Selected	TP	FP
	Not selected	FN	TN

- Main measures

Precision (P): proportion of selected items that are correct

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

Recall (R): proportion of corrrect items that are selected

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

F_measure: a combined

$$F = \frac{1}{\theta \times \frac{1}{P} + (1 - \theta) \times \frac{1}{R}}$$

Results and conclusions

- Character embeddings are more effective than raw character representations
  - No significant differences between CNN and LSTM based representations
- Word embeddings are more effective than raw words
  - LSTM-based representations are more effective than CNN-based representations
  - Neural inference based on embeddings (eg. perceptron)
- Softmax-layer is more effective than CRF for prediction output

Results (F1-score)			NER			
			WLSTM+CRF	WLSTM	WCNN+CRF	WCNN
Nochar	Literature		90.10 (H-15)* 90.20 (L-16) 90.43 (S-17)*	87.00 (M-16) 89.34 (S-17)*	89.59 (C-11)* 90.54 (S-17)*	89.97 (S-17)*
	Ours	Max	89.45	88.57	88.90	88.56
		Mean±std	89.31±0.10	88.49±0.17	88.65±0.20	88.50±0.05
CLSTM	Literature		90.94 (L-16) 91.20 (Y-17)‡	89.15 (L-16)	–	–
	Ours	Max	91.20	90.84	90.70	90.46
		Mean±std	91.08±0.08	90.77±0.06	90.48±0.23	90.28±0.30
CCNN	Literature		90.91±0.20 (C-16) 91.21 (M-16) 90.87±0.13 (P-17)	89.36 (M-16)	–	–
	Ours	Max	91.35	90.73	90.43	90.51
		Mean±std	91.11±0.21	90.60±0.11	90.28±0.09	90.26±0.19

ESILV

Datesets

Corpus	Year	Text source	#Entity categories	URL
MUC-6	1995	Wall Street Journal	7	https://catalog.ldc.upenn.edu/LDC2003T13
CoNLL03	2003	Reuters News	4	https://www.clips.uantwerpen.be/conll2003/ner/
ACE	2000- 2008	Transcripts, news	7	https://www.ldc.upenn.edu/collaborations/past-projects/ace
OntoNotes	2007- 2012	Magzines, news, web	18	https://catalog.ldc.upenn.edu/LDC2013T19
WikiGold	2009	Wikipedia	4	https://figshare.com/articles/Learning_multilingual_name d_entity _recognition_from_Wikipedia/5462500
WiNER	2012	Wikipedia	4	http://rali.iro.umontreal.ca/rali/en/winer-wikipedia-for-ner
BC5CDR	2015	PubMed	3	http://bioc.sourceforge.net/
DFKI	2018	Business news and social media	7	https://dfki-lt-re-group.bitbucket.io/product-corpus/

Relation Extraction (RE)

- Relation extraction (RE) is the process of identifying and classifying semantic relationships between NE in text.
  - Example: From "Barack Obama was born in Hawaii", extract the relation **BornIn(Barack Obama, Hawaii).**
- Both RE and NER are essentially annotation process.
  - They share common learning methodologies and feature extraction.
    - We can consider the relations as specific annotation labels
  - Differences:
    - RE often requires additional features that capture the connection between pairs of entities.
    - The output of RE are structured relations, in form of triples, while NER outputs only a sequence of lables

ESILV

Knowledge Graph building

ESILV

Process of knowledge base/graph building

1. Representing Extracted Information as Triples (RDF triples) ✅
2. Defining an Ontology(Using RDF, RDFS, OWL, etc.) ✅
3. Data Integration and Normalization
   - Entity linking with external knowledge bases
4. Storing the Knowledge Graph
   - Graph databases and RDF stores
5. Populating the Knowledge Graph
   - Write scripts or use ETL tools to convert NER and RE into triples
6. Querying and Enriching the Graph
   - SPARQL Query ✅

ESILV

Data Integration and Normalization

- Data integration and normalization is a critical step in building a high-quality knowledge graph.
- Identical entity often have variations
  - e.g., "IBM," "International Business Machines," "IBM Corp."
- Objective: resolve these variations and map them to unique, canonical representations.

ESILV

Entity Resolution / Linking

- Purpose: Merge different mentions of the same entity into a single canonical entity.
- Techniques:
  - **String Similarity & Matching:**
    - Use similarity measures (like Levenshtein distance or **cosine similarity** on **word embeddings**) to identify similar entity mentions.
  - **Contextual Clues:**
    - Analyze surrounding text or use features like entity type, part-of-speech tags, and syntactic context to disambiguate entities.
  - **Knowledge Base Mapping:**
    - Use external knowledge bases (e.g., DBpedia, Wikidata, or Freebase) to link extracted entities to canonical identifiers.
    - Tools like **DBpedia Spotlight** or **TAGME** can automate this mapping.

ESILV

Example: Using DBpedia Spotlight

- **DBpedia Spotlight** is an open-source tool that identifies and links named entities in text to their corresponding entries in DBpedia.
- Use API request
  - Send your text to Dbpedia Spotlight API with HTTP GET (e.g., "Barack Obama was the president. Obama was born in Hawaii. President Obama...")
  - Make sure to set the appropriate HTTP headers (e.g., Accept: application/json).

```
curl 'https://api.dbpedia-spotlight.org/en/annotate?text=Barack%20Obama%20was%20the%20president.%20Obama%20was%20born%20in%20Hawaii.%20President%20Obama...&confidence=0.5' -H "Accept: application/json"
```


ESILV

```
{
    "@text": "Barack Obama was the president. Obama was born in Hawaii. President Obama...",
    "@confidence": "0.5",
    "@support": "0",
    "@types": "",
    "@sparql": "",
    "@policy": "whitelist",
    "Resources": [
        {
            "@URI": "http://dbpedia.org/resource/Barack_Obama",
            "@support": "28423",
            "@types": "Http://xmlns.com/foaf/0.1/Person,Wikidata:Q82955,Wikidata:Q729,Wikidata:Q5,Wikidata:Q215627,Wikidata:Q19088,DUL:NaturalPerson,Schema:Person,DBpedia:Species,DBpedia:Person
            "@surfaceForm": "Barack Obama",
            "@offset": "0",
            "@similarityScore": "0.9999070831866319",
            "@percentageOfSecondRank": "6.900077157434426E-5"
        },
        {
            "@URI": "http://dbpedia.org/resource/Barack_Obama",
            "@support": "28423",
            "@types": "Http://xmlns.com/foaf/0.1/Person,Wikidata:Q82955,Wikidata:Q729,Wikidata:Q5,Wikidata:Q215627,Wikidata:Q19088,DUL:NaturalPerson,Schema:Person,DBpedia:Species,DBpedia:Person
            "@surfaceForm": "Obama",
            "@offset": "32",
            "@similarityScore": "0.9995323303363935",
            "@percentageOfSecondRank": "3.410243018851691E-4"
        },
        {
            "@URI": "http://dbpedia.org/resource/Hawaii",
            "@support": "36120",
            "@types": "Wikidata:Q3455524,Schema:Place,Schema:AdministrativeArea,DBpedia:Region,DBpedia:PopulatedPlace,DBpedia:Place,DBpedia:Location,DBpedia:AdministrativeRegion",
            "@surfaceForm": "Hawaii",
            "@offset": "50",
            "@similarityScore": "0.9996913972780808",
            "@percentageOfSecondRank": "9.235270515071463E-5"
        },
        {
            "@URI": "http://dbpedia.org/resource/Barack_Obama",
            "@support": "28423",
            "@types": "Http://xmlns.com/foaf/0.1/Person,Wikidata:Q82955,Wikidata:Q729,Wikidata:Q5,Wikidata:Q215627,Wikidata:Q19088,DUL:NaturalPerson,Schema:Person,DBpedia:Species,DBpedia:Person
            "@surfaceForm": "President Obama",
            "@offset": "58",
            "@similarityScore": "0.999803937517009",
            "@percentageOfSecondRank": "1.9610093095780894E-4"
        }
    ]
}
```


Once we have the Dbpedia URI, we can map it into a unique identifier (Barack Obama: Wikidata Q76)

Integration into the Knowledge Graph/Base

- Once data is normalised, we can build the knowledge graph.
- Use RDF libraries, such as Apache Jena (in Java) and RDFLib (in Python). The knowledge graphs are stored via rdf files.
- The knowledge graph can be stored using Graph databases, like Neo4j, GraphDB, etc. They allow to store and manage the knowledge graph with database management system feature, such as optimised query, back-up, conflict management, distributed storage, etc. (You will have graph mining module with Prof. Nicolas Travers)

Summary

- Brief introduction to web crawling/scrapping
- Information Retrieval from text
- Knowledge graph construction pipeline
  - Data collection
  - Preprocessing
  - NER/RE
  - Knowledge graph construction

ESILV

References

- Web Crawling by Filippo Menczer, Indiana University School of Informatics
- Information Extraction and Analysis from Text Media, by Lynda Tamine-Lechani, University Toulouse III, Paul Sabatier
- Web Data Mining, by Bing Liu, University of Illinois at Chicago
- Introduction to Information Retrieval, by Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Combridge University Press, 2008

ESILV