Information Extraction

Oana Balalau

Inria, Ecole Polytechnique
oana.balalau@inria.fr

28/03/2025

Motivation: Machine Knowledge

Much of the knowledge we have as humans is stored as **text**.
Many of your favorites books are just collections of sentences!
Can we transform these sentences in edges in a graph? Yes!
(almost)

Motivation: Machine Knowledge

Data models such as RDFs are easy to use by algorithms, especially compared to text. Constructing very large RDFs is time consuming.

Information extraction methods enable extracting structured content from digital text documents.

Structured content can be used to automatically populate knowledge bases / knowledge graph (KB, stored as RDF graphs): machine-readable facts about the real world.

Information extraction

Information Extraction: extract structured data from unstructured (typically text) or semi structured machine-readable information (e.g. tables, json etc).

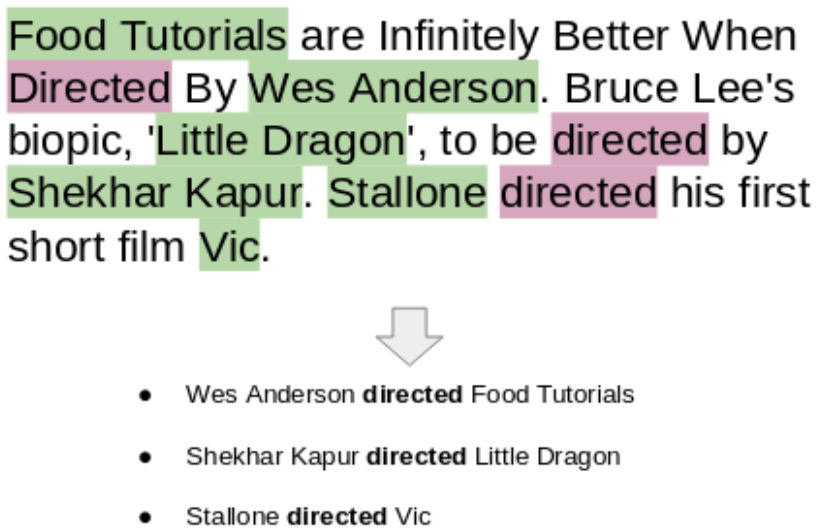


Figure: Extracting triples of the form (subject, predicate, object)

- Information Extraction $\subsetneq$ Natural Language Processing.

Information extraction

Figure: We can extract triples of the form (subject, predicate, object) from infoboxes (Yago KB)

What is a Knowledge Base?

A **knowledge base** (KB) is a collection of structured data about entities and relations [Weikum et al., 2020]. A KB is stored in the RDF format.

- An **entity** is any abstract or concrete object of fiction or reality (e.g. house elf).
- An **individual entity** is an entity that can be uniquely identified against all other entities (e.g. Dobby the house elf).
- An **identifier for an entity** is a string of characters that uniquely denotes the entity (URI - Uniform Resource Identifier).
- A **mention of an entity** in a data source (including text) is a string (including acronyms and abbreviations) intended to denote an entity (e.g. *Dobby brought the potion.*).

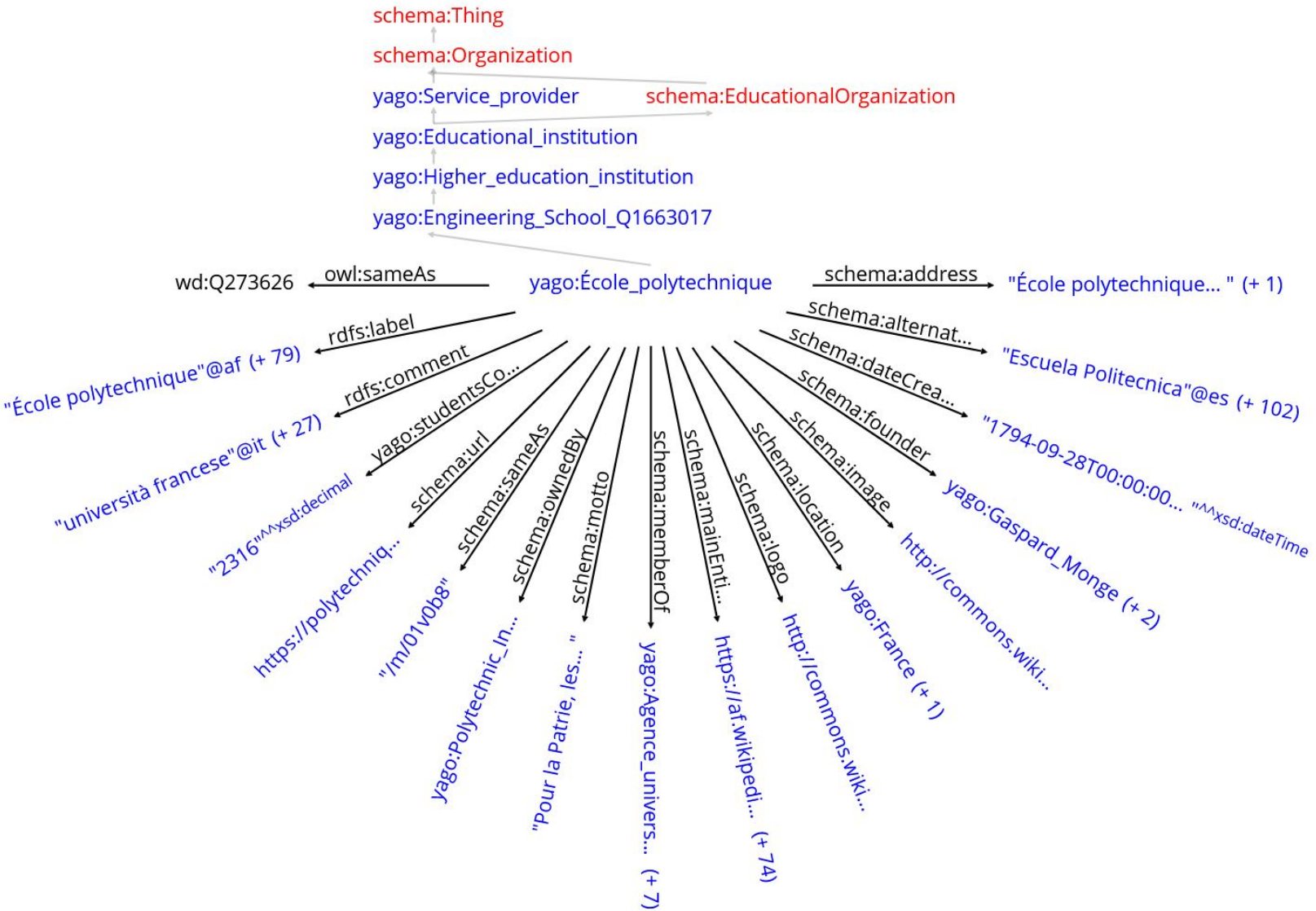
What is a Knowledge Base?

- A **class, or interchangeably type**, is a named set of entities that share a common trait. An element of that set is called an instance of the class (e.g. Dobby).
- Class A is a **subclass** of (is subsumed by) class B if all instances of A must also be instances of B (presidents subclass of people).
- A **taxonomy** is a directed acyclic graph, where the nodes are classes and there is an edge from class X to class Y if X is a direct subclass of Y.
- A **relation** or relationship for the instances of classes $C_1, ..., C_n$ is a subset of the Cartesian product $C_1 \times ... \times C_n$, along with an identifier (i.e., unique name) for the relation.

What is a Knowledge Base?

- The **schema of a KB** consists of
  - a set of types, organized into a taxonomy, and
  - a set of property types – attributes and relations – with type signatures.
- The **RDF model** restricts the three roles in a subject-predicate-object (SPO) triple as follows:
  - S must be a URI identifying an entity,
  - P must be a URI identifying a relation, and
  - O must be a URI identifying an entity for a relationship between entities, or a literal denoting the value of an attribute
- Each entity, class and property in a KB is **canonicalized** by having a unique identifier.

Ecole polytechnique in the Yago KB



Creating a Knowledge Base/Knowledge Graph

Creating a KB

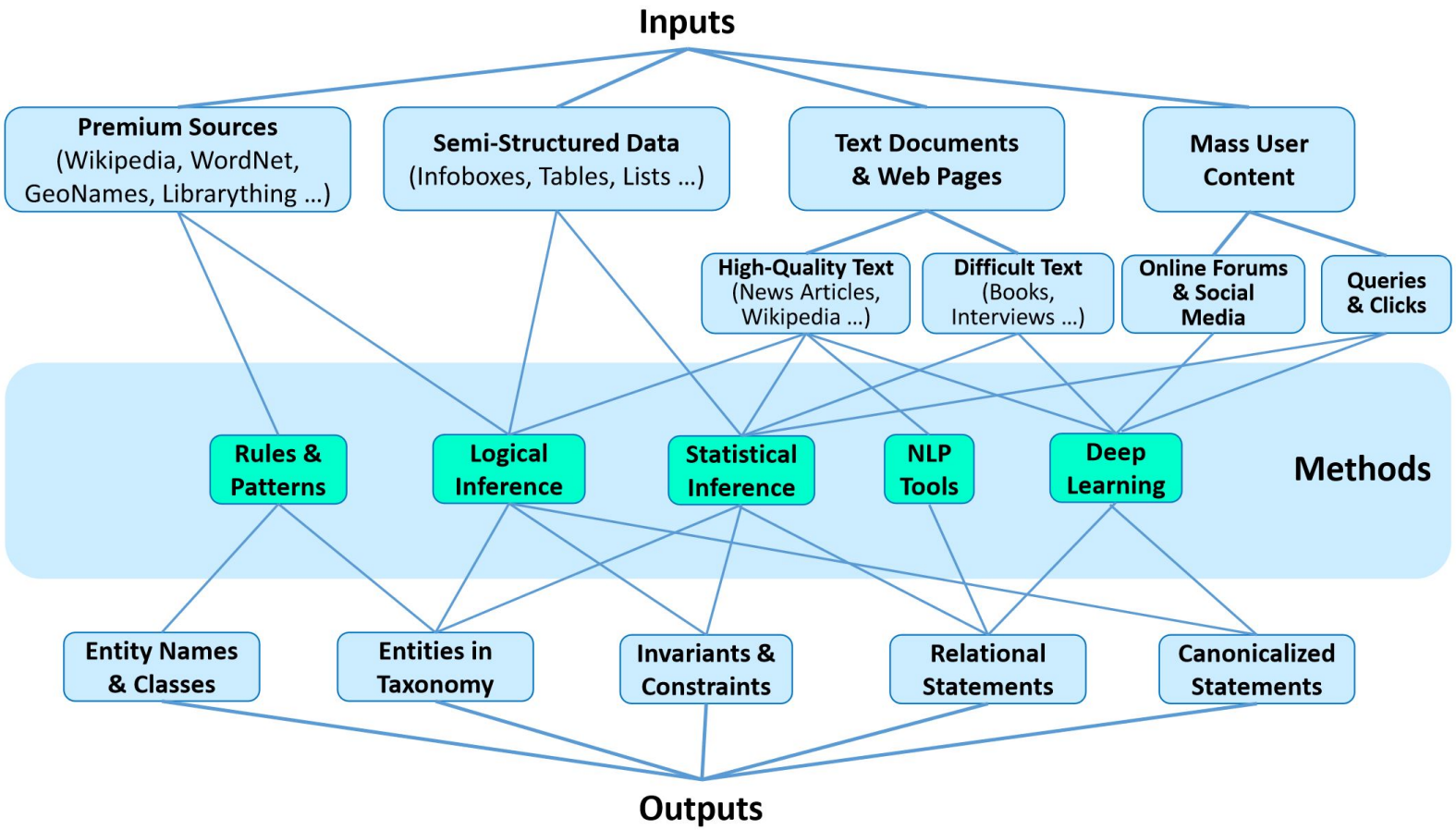


Figure: Creation of a KB, example from [Weikum et al., 2020]

Information extraction tasks

Auxiliary tasks:

- sentence segmentation
- tokenization
- lemmatization
- part-of-speech tagging
- dependency parsing

Tasks useful for creating a KB:

- named entity recognition
- coreference resolution
- entity canonization (named entity linking)
- relation extraction / fact extraction
- relation canonization

Sentence segmentation

Given the text: *Ms. Smith said that spring is a wonderful season. What do you think?*

- punctuation (".?!") based: {"Ms.", "Smith said that spring is a wonderful season.", "What do you think?"}
- punctuation with list of abbreviations: {"Ms. Smith said that spring is a wonderful season.", "What do you think?"}

However, this approach will fail on informal text, such as tweets:
you said spring is wonderful jane hates spring I prefer winter
There are syntactic patterns: the subject will start a sentence.

Tokenization

Given the sentence: *Spring is a wonderful season, isn't it?*
we can have different tokenization strategies:

- Space-based: {"Spring", "is", "a", "wonderful", **"season,"**, "isn't", **"it?"**}
- Space and punctuation: {"Spring", "is", "a", "wonderful", "season", ",", **"isn"**, **""'**, **"t"**, "it", "?"}
- Space, punctuation and rules: {"Spring", "is", "a", "wonderful", "season", ",", "is", "n't", "it", "?"}

In French we have *la, le* which can become *l'*.
In German compound nouns are written without spaces, e.g.
Computerlinguistik is computational linguistics.
In Chinese and Japanese there is no space!

Lemmatization

When extracting information from text, we want information that is in a standard format, for example:

am, are, is $\Rightarrow$ be
car, cars, car's, cars' $\Rightarrow$ car
walk, walking, walked $\Rightarrow$ walk

The standard form of a word is the **lemma** and the method is **lemmatization.**
One simple approach is to search for the lemma of a word in a dictionary, such as **WordNet**.

WordNet

WordNet: lexical database of semantic relations between words[1].

Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept.

Synsets are linked by relations: is a, part of, antonym.

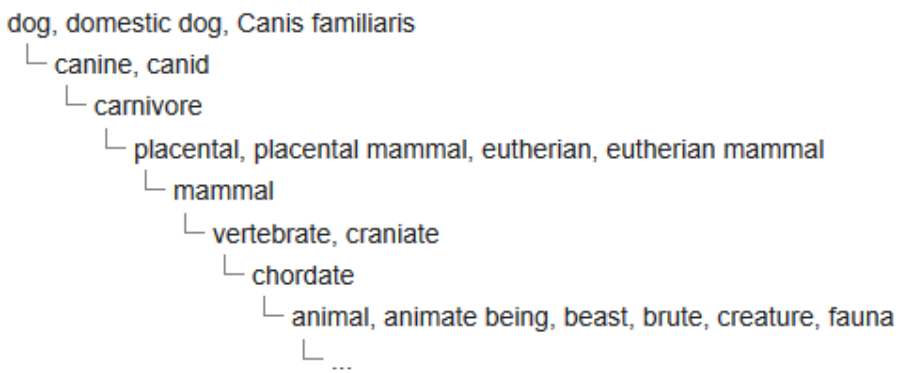


Figure: The IS A / hypernym relation

Wordnet: simulate organization of human verbal memory.

[1] To test it: `http://wordnetweb.princeton.edu/`

Part-of-speech tagging (POS)

Figure: We label text with POS tags, using `https://parts-of-speech.info/`

Part-of-speech tagging

She will **race** you for it.
When will the **race** start?

On average a word can have 2 part of speech tags.
90% accuracy by choosing the most frequent tag for a word.

For even better performance we can use the context:

1. **rule-based**: create a large database of hand-written disambiguation rules, e.g. specify that a word is a noun rather than a verb if it follows a determiner (e.g. *the race*).

2. **probabilistic**: resolve tagging ambiguities by using a training corpus to compute the probability of a given word having a given tag in a given context.

Dependency parsing

Dependency parsing is the process of determining the syntactical structure of a sentence. Syntax refers to the arrangement of words and phrases in a sentence, finding multiple words that act as a single unit.

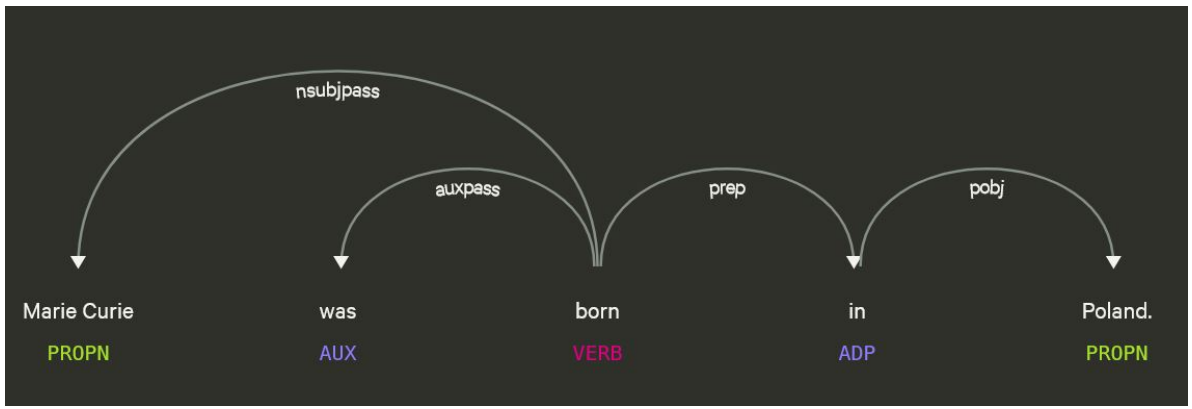


Figure: We label text with dependency tags tags, using `https://demos.explosion.ai/displacy`

The tag explanation is here:
`https://universaldependencies.org`

Information extraction tasks

Auxiliary tasks:

- sentence segmentation
- tokenization
- lemmatization
- part-of-speech tagging
- dependency parsing

Tasks useful for creating a KB:

- named entity recognition
- coreference resolution
- entity canonization (named entity linking)
- relation extraction / fact extraction
- relation canonization

Named entity recognition (NER)

A named entity refers to a **person, location, organization, artifact** with a name: *Paris, Victor Hugo, The Hunchback of Notre Dame*.

Named entity recognition (NER) is the task of finding entity names in a text. Often it involves also specifying the class of an entity (e.g. person, location, organization, etc.)

The first word in a sentence is capitalized.
Capital Letters can be used to mark Importance.
in informal text, such as tweets, there is no rule

Named entity recognition (NER)

First solution: all the words starting with capital letters are entities. Problems: beginning of sentences, prepositions in names (The Hunchback of Notre Dame), informal text with no/poor capitalization.

Second solution: have a dictionary of named entities. Problem: will not find entities that do not belong to the dictionary.

Third solution: Annotate text with the entities it contains and train a neural network.

A bit on regular expressions

What if we have identified words following a pattern and we would like to automatically extract them?
E.g. names, dates (DD-MM-YYYY), email addresses, or high level annotations such as POS tags.

A regular expression finds words over an alphabet (the alphabet can be the set of unicode characters, or a set of labels).
A language is a set of words: for example a, aa, aaa, aaaa, etc.
A regular expression describes a language.

A bit on regular expressions

$L1 = \{, a, aa, aaa, ...\} \rightarrow a*$
'*' (asterisk) matches zero or more of the preceding character.

$L2 = \{ab, abc, abcc, abccc, ...\} \rightarrow abc*$

$L3 = \{, ab, abab, ababab, ...\} \rightarrow (ab)*$
parentheses are used to group regex letters together, () is a group

$L4 = \{0, 1, 2, 3, 1, 5, 6, 7, 8, 9\} \rightarrow (0|1|2...|9) = [0-9]$

$L5 = \{Albert\ Einstein, Harry\ Potter, ...\} \rightarrow [A-Z][a-z]+\ [A-Z][a-z]+$
'+' (plus) matches one or more of the preceding character

$L6 = \{a, ab, abb, bbba, aaabbbab, ababa, ...\} \rightarrow (a|b)+$

$L7 = \{ac, b, aa, bbc, aaa, bbb, ...\} \rightarrow (a+|b+)c?$
'?' (question mark) means match zero or one instance of this character

Efficiently finding words in a data structure: Trie

A trie (pronounced as tree), also called a digital tree or a prefix tree, is a tree where nodes are boolean and edges are labeled with letters.

A string **S** is contained in the trie if S denotes a path from the root to a node labelled true.

Example: Ada, Adams, Adora.

To add a node that is a prefix of an existing string, switch the node to true. To add a string with no common prefix, create a new branch.

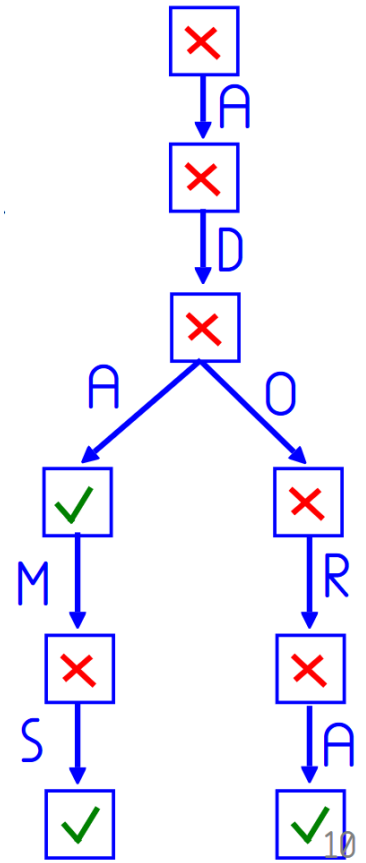


Figure: Example from [Suchanek, 2021].

Efficiently finding words in a data structure: Trie

To find all the named entities in a text: from every character in the text advance until you find a true node or until you cannot continue in the trie.

Adams adores Adora.

Running time: O(textLength X maxDepthTrie)

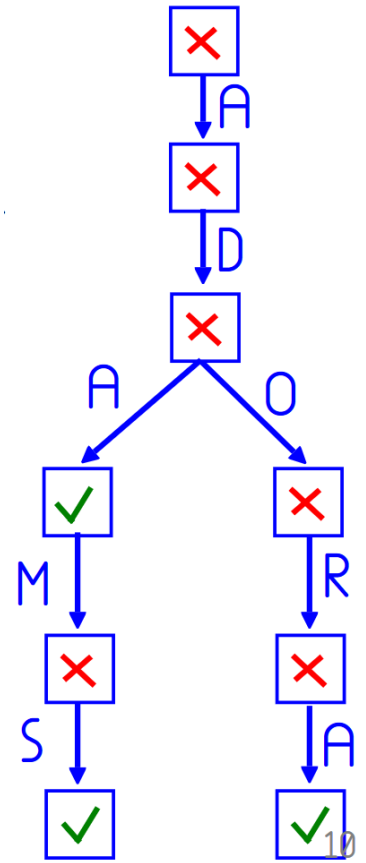


Figure: Example from [Suchanek, 2021].

Named Entity Recognition

Best performance on the task is given by neural models, as high as 94% F1 score[2].

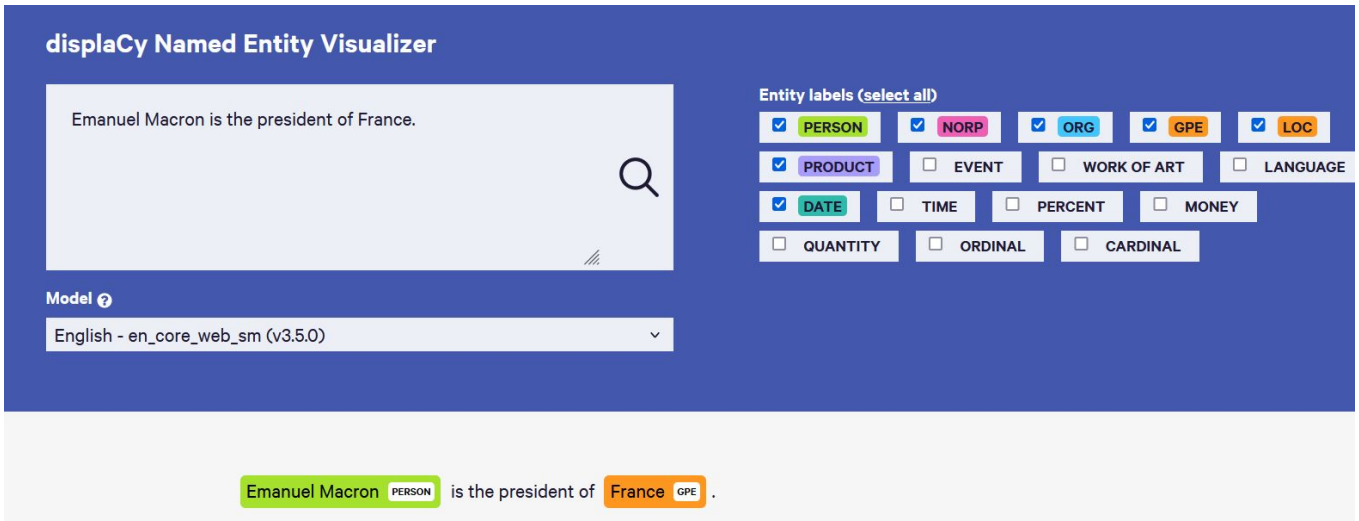


Figure: We label text with NER tags, using `https://demos.explosion.ai/displacy-ent`

[2] `https://paperswithcode.com/task/named-entity-recognition-ner`

Coreference resolution

Coreference resolution is the task of finding all expressions that refer to the same entity in a text.

Examples:

- anaphora: *The sun is so strong.***It** *can burn you.*
- cataphora: *Look at* **them**, *Ana and John are making the fire.*
- coreferring noun phrases: *Look at* **those kids**, *Ana and John are making the fire.*

Coreference resolution

How to address coreference resolution:

- **closeness**: the closest entities are more likely to be referred.
- **grammatical role**: entities in the subject position are more likely to be referred, *The sun is so strong.* **It** *can burn you.*
- **verb semantics**: Ana did not pick the phone because she was tired. Ana did not pick the phone because it was not ringing.
- considering **gender** (female, male), **count** (single or plural) and **person** (1st, 2nd or 3rd person)

Using these features we can have a rule based approach or a supervised approach.

Canonization of entities: named entity linking

Named entity linking / disambiguation is the task of assigning to a entity in a text (called a **mention**) a unique identifier, usually by linking it to a standard data source, such as KB.

Europe was kidnapped by Zeus. $\Rightarrow$ Running NERC will give us that Europe is a person $\Rightarrow$ If in our KB we have only two options, the location and the mythological character, we match it to the latter.

There was a heavy rain in Paris. $\Rightarrow$ Simple strategy: match the mention to the corresponding KB entity that is the most complete.

Paris (Q90)
capital and largest city of France
415 statements, 329 sitelinks - 04:51, 2 April 2021

Paris (Q830149)
county seat of Lamar County, Texas, United States
40 statements, 52 sitelinks - 15:42, 2 April 2021

Named entity recognition and classification (NERC)

Named entity recognition and classification: also known as **entity typing**.

Task: give a named entity, assign a type to it.
Types: **person, place, location, event**.

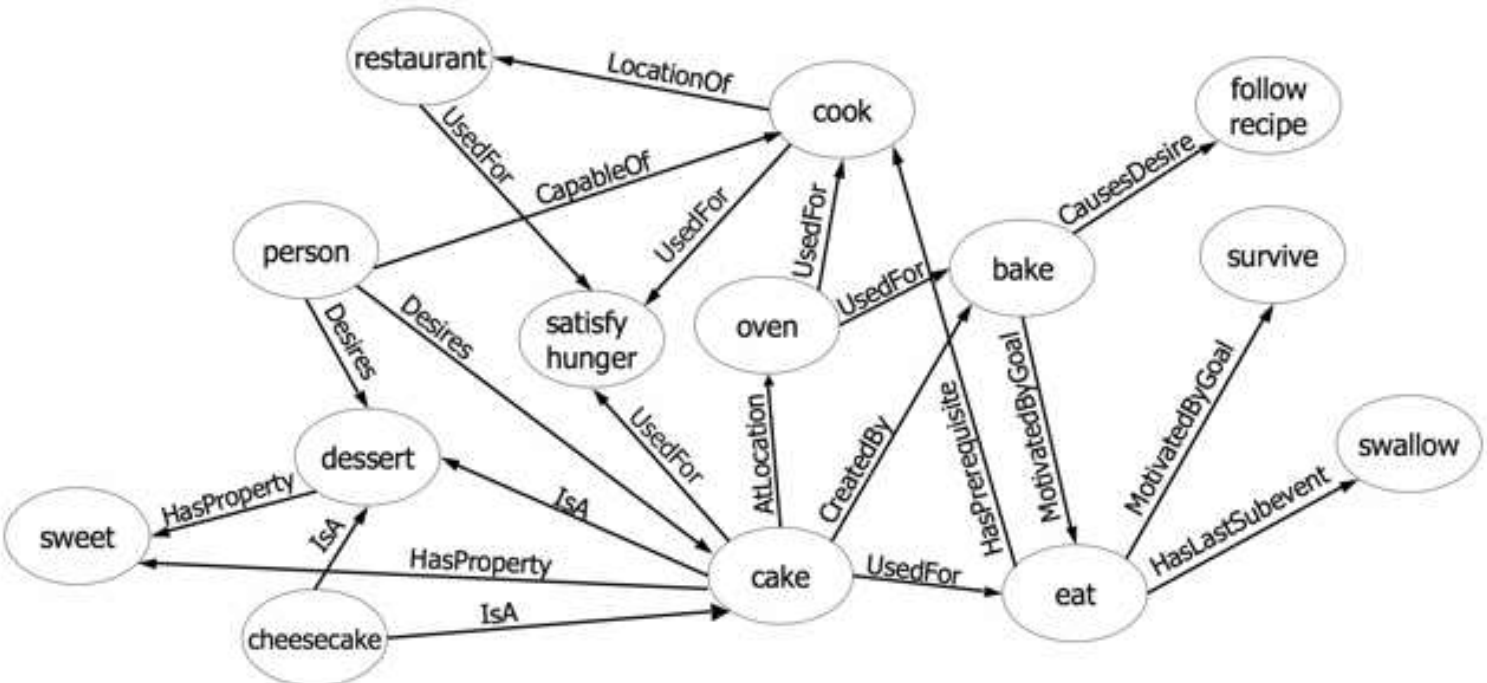
We can also have unnamed entities: **statistical entities, food, dates etc.**.

Named vs unnamed entities: proper nouns vs simple nouns or noun phrases.

Common sense knowledge graphs

Common sense KBs: Cyc, Concept Net, Quasimodo, etc.

Motivation: common sense is something that we gain by living and interacting with our environment, which is not often expressed in text. Hence, a machine cannot learn it.



Named entity recognition and classification (NERC)

Hearst Patterns: Consider patterns that co-occur with input entity and the desidered output class. E.g. "writers such as Jon Steinbeck": Jon Steinbeck is a writer.

1. **Input:** Start with a small set of trusted entities of a given class: $S = (Jon\ Steinbeck, Terry\ Pratchett)$
2. **Output:** Patterns for this class, and new entities of the class
3. Initialize P with empty set or with pre-specified patterns
4. 1. Pattern discovery:
   1. identify co-occurring phrases with entity $e \in S$ in web corpus;
   2. generalize phrases into patterns ("writers such as X");
   3. add frequent patterns;
5. 2. Statement expansion:
   1. identify co-occurring entities with patterns $p \in P$ in web corpus;
   2. add frequent entities to S
6. Repeat steps 4 and 5 until no new patterns can be learnt.

0	sentence:	Singers like Elvis Presley were kings of rock'n'roll.
1.1	new pattern:	*singers like* $X
1.2	sentence:	Singers like Nina Simone fused jazz, soul and gospel.
	new entity:	`Nina Simone`
2.1	sentence:	Nina Simone's vocal performance was amazing.
	new pattern:	$X *'s vocal performance*
2.2	sentence:	Amy Winehouse's vocal performance won a Grammy.
	new entity:	`Amy Winehouse`
	sentence:	Queen's vocal performance led by Freddie Mercury ...
	new entity:	`Queen`
3.1	sentence:	The voice of Amy Winehouse reflected her tragic life.
	sentence:	The voice of Elvis Presley is an incredible baritone.
	new pattern:	*voice of* $X
3.2	sentence:	The melancholic voice of Francoise Hardy ...
	new entity:	`Francoise Hardy`
	sentence:	The great voice of Donald Trump got loud and angry.
	new entity:	`Donald Trump`
...	...	...

Figure: Example from [Weikum et al., 2020]

Canonization of entities: named entity linking

Last time we have seen other techniques:

1. String similarity ⇒ President Macron vs Emmanuel Macron;
2. Context / neighbourhood similarity
   - mention context: window of tokens around the mention, the sentence / paragraph / document containing it.
   - entity in KB context: its label, hand selected attributes or all attributes

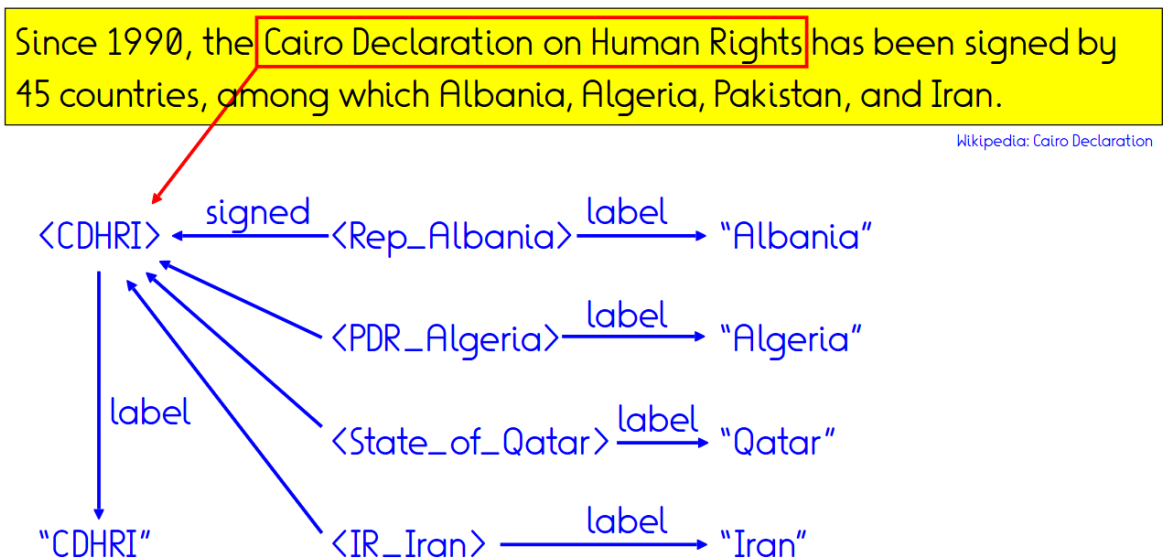


Figure: Example from [Suchanek, 2021].

Relation extraction / fact extraction

Relation extraction, or fact extraction, is the task of extracting triples of the form *subject*, *predicate*, *object* from a sentence. The subject should be a named entity.

Paris is the capital and most populous city of France, with an estimated population of 2.175.601 residents as of 2018, in an area of more than 105 square kilometres.

Facts/ KB triples:
Paris, capitalOf, France
Paris, hasPopulation, 2.175.601
Paris, hasArea, 105 km^2

Relation extraction / fact extraction

Two types of extraction:

1. extraction of a fixed and known number of relations.
2. extraction without any knowledge of the relations to extract, also called **open information extraction**. The subject is not necessarily an entity.

1. Relations are known. We can use extraction patterns: *X wrote Y*, *X is the author of Y* etc. We match the pattern to a sentence and we find a fact: *(John Steinbeck, wrote, East of Eden.)* How can we find many extraction patterns?

Relation extraction / fact extraction

1. Relations are known. DIPRE [Brin, 1998] algorithm: for a given input relation **r** :

1. Start with a small set of trusted pairs (subject, object) that are in relation **r**: *(John Steinbeck, East of Eden)*
2. Find occurrences of these pairs in sentences.
3. Identify generalized patterns from the sentences fetched in step 2: *X is author of Y, X is the creator of Y*.
4. Use these identified patterns to find more pairs (subject, object): *(Jane Austen, Emma)*.
5. Repeat steps 2 to 4 until no new patterns can be learnt.

At the end we will have a set of facts and patterns for a given relation r.

Relation extraction / fact extraction

2. Relations are not known. The ReVerb algorithm [Fader et al., 2011] has two steps: relation extraction and argument extraction.

A relation is of the form **V|VP|VW*P** where
V=(aux verb|verb) particle? adverb?
W=(noun|adj|adv|pron|det)
P=(prep|particle)

Example: verb (*walks*), a verb followed by a preposition (*located in*), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (*has atomic weight of*).

The arguments of the relations are the nearest noun phrases to the left and to the right of the relation (My best friend walks every day.).

Canonization of relations via clustering

After running the ReVerb algorithm we will get many predicates that express the same meaning:

- locatedIn: is situated in, is part of, is located in, belong to
- spouse: is married with, is the spouse of, is the husband of, is the wife of
- etc..

How to group together relations that have the same semantic meaning?

Canonization of relations via clustering

Features for clustering (grouping) relations:

- String similarity of predicate phrases, using embeddings (is married with, is the spouse of)
- Overlap between the entities that are in the arguments of two predicates (B.Obama and M.Obama are both in the arguments of is married with, is the spouse of).
- Type signature similarity between phrases (subject is person, object is person)
- Context similarity based on the sentences or passages from where the triples were extracted.

Evaluating the quality of a KB

Precision, recall, F1

Precision and recall for:

- for a set of entities that populate a given class;
- for all entities of a given type;
- for the set of classes known for a given entity;
- for the set of property statements about a given entity;

We assume the KB contains a set of statements S we want to evaluate, and a ground-truth set GT.

$$\text{Precision}(S) = \frac{|S \cap GT|}{|S|}$$

$$\text{Recall}(A) = \frac{|S \cap GT|}{|GT|}$$

Creating the ground truth

Creating a ground truth: depending on what we evaluate, we would need a very comprehensive set (e.g. all the songs of a singer, all the people that are singers).

In practice, we often resort to proxies for the grounth truth.

Precision: We select a (random) subset of elements from the KB, and for each of them, we decide if it is correct or not.

Recall: Sampling is not ideal, as the goal is capturing all knowledge of interest. We instead collect a small but complete ground-truth set, for example, all songs by a singer.

Knowledge Base Completeness

The Open-World Assumption (OWA) postulates that if a statement is not in the KB, it may nevertheless be true in the real world. Its truth value is unknown. In other words, absence of evidence is not evidence of absence [Weikum et al., 2020].

The Local Completeness Assumption (LCA), aka. Partial Completeness Assumption or Local Closed-World Assumption, for entity S in the KB asserts that if, for some property P, the KB contains at least one statement ¡S, P,O¿, then it contains all statements for the same S and P. For example, if we know one of a person's children, then we know all of them; S is complete with regard to P [Weikum et al., 2020].

Summary

Information extraction offers tools to **construct graphs from textual data** using named entity recognition and relation extraction.

Two broad approaches for a IE task:

- Knowledge engineering: find important features that can help constructing rules or patterns. Features should be given by domain experts. **Very hard to cover everything, language is rich and ambiguous.**
- Statistical methods that learn from annotated corpora. Require a large dataset with human annotations. **These techniques perform the best.**

References I

Brin, S. (1998).
Extracting patterns and relations from the world wide web.
In International workshop on the world wide web and databases, pages 172–183. Springer.

Fader, A., Soderland, S., and Etzioni, O. (2011).
Identifying relations for open information extraction.
In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Suchanek, F. (2021).
Information Extraction.
`https://suchanek.name/work/teaching/topics/`.
[Online; accessed April 2021].

Weikum, G., Dong, L., Razniewski, S., and Suchanek, F. M. (2020).
Machine knowledge: Creation and curation of comprehensive knowledge bases.
CoRR, abs/2009.11564.