Information Extraction

Andrea Esuli

Sequence labeling

Many NLP and information extraction tasks are focused on determining some properties of interest inside a piece of text:

- PoS of every word
- Syntactic role of every word
- Determining if a word, or a sequence of words, identifies a certain type of information, e.g., the name of a person/location/brand
- Infer other properties, e.g, the unit of measure of a number, "I am 1.80" vs "I am 42"
- Link pieces of text that are related, e.g., "*Andrea* is a researcher, *he* is from Pisa"
- Link a piece of text to element of a knowledge base/ontology

Sequence labeling

In these tasks, a document is no more an atomic entity, but it is processed as a sequence of token.

There is not a one-to-one relation between document and output, such as in classification.

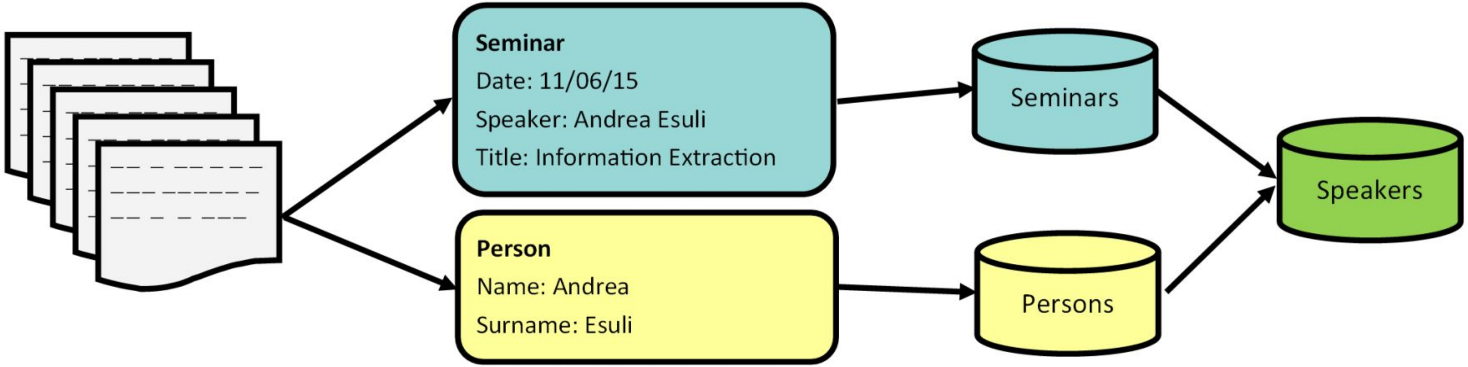
We can extract from text a variable amount of information, depending on its length, but also on its relevant to the specific tasks we apply to it.

Every token is the subject of the analysis and their order in text obviously play a relevant role in the outcome of the analysis.

Information Extraction

Information Extraction (IE) is about extracting structured information from unstructured or partially structured text.

IE is a step toward higher level (semantic) representation of knowledge with respect to classical IR (e.g., web search).



Information Extraction

Two key IE tasks:

- Named entities recognition

  `Andrea Esuli is a researcher as ISTI-CNR`

  p_1= Person(*Andrea Esuli*)

  o_1= Organization(*ISTI-CNR*)

- Relation extraction

  `Andrea Esuli is a researcher at ISTI-CNR`

  r_1= Role(*researcher*,p_1,o_1)

Named Entity Recognition

Named Entity Recognition (NER) is the problem of identifying pieces of text that refer to elements belonging to predefined categories such as:

- Persons

  `Andrea Esuli, Mario Rossi, Rossi, President of USA, President`
- Organizations

  `Inter, Milan, Roma, Lazio`
- Locations

  `Milan, Pisa, via Garibaldi, Lazio, Tuscany, Arno, Tirreno`
- Temporal expressions

  `July 3, Friday, today, last century, the '60, for an hour`
- Quantities

  `one kilogram, one kilo, 2 tera, a quarter, a dozen`

Named Entity Recognition

The problem can be split into subproblems:

- entity spotting:

    I saw *Andrea Esuli* riding his bike.

- entity classification:

    Andrea Esuli → Person

- entity identification (a.k.a. entity linking, wikification):

    Andrea Esuli → http://www.esuli.it (URI)

The first two steps are usually performed together.

NER using Rules

Lexicons (dictionaries, gazetteers, ontologies) play a relevant role in *entity spotting*.

Rules are usually hand-made, and have the form of patterns and properties that have to match the context of the NE to have a recognition.

Example of extraction rule, adapted from ANNIE

```
Rule: isFemale({
    Lookup.class == female_person_first_name,
    Lookup.ontology == "gate:/creole/ontology/demo.daml"
}):person
```

NER using ML

Machine learning-based IE usually translates the extraction problem into *a word classification problem*.

`Barack Obama flew to Rome last week`

$[\texttt{Barack Obama}]_{\texttt{per}}$ $\texttt{flew to}$ $[\texttt{Rome}]_{\texttt{loc}}$ $[\texttt{last week}]_{\texttt{time}}$

A *binary word classifier* is learned for each type of recognized entity.

- A classifier classifies every word as representing or not an entity.
- Depending if annotation can overlap or not, i.e., a word can have only one label type or more, the output of the classifiers is combined in a *single-label* classification or a *multi-label* one.

NER using ML

Each word is represented by features that capture its morphologic, syntactic , and semantic properties…

```
r(Barack)=['Barack', 'barack', firstCap, mixCase, NNP, male …]
```

…and those of its context, usually defined as a set of preceding and following words.

The vector that represents a word is thus the concatenation of the features that define the observed word and those of the context words (taking into account their relative position):

$$v(w_i) = r(w_i) + r(w_{i-1}) + r(w_{i-2}) + r(w_{i+1}) + r(w_{i+2})$$

NER using ML

Once we assign a proper representation (e.g., probabilistic or vectorial) to every element that is the object of the annotation, traditional machine learning methods seen for text classification can be applied to IE.

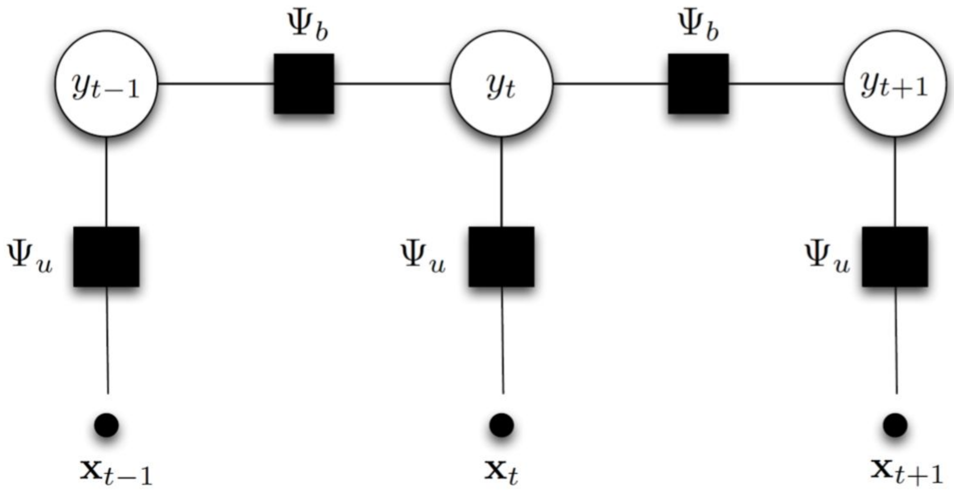
Neural networks (recurrent and attention models) have also found successful application to tagging, IE, Entity linking.

The linear structure of text can further exploited (in addition to features) by using *graphical models*.
A graphical model is a *probabilistic model* that represents the dependencies between the observed objects with a graph.

NER using graphical models

Conditional Random Fields (CRFs) are a kind of **probabilistic graphical models** that explicitly model dependencies among variables of the problem, including those that happen among labels.

A *linear chain* CRF determines the labeling of a piece of text on all the words at the same time, by maximizing the labeling probability of the whole sequence.

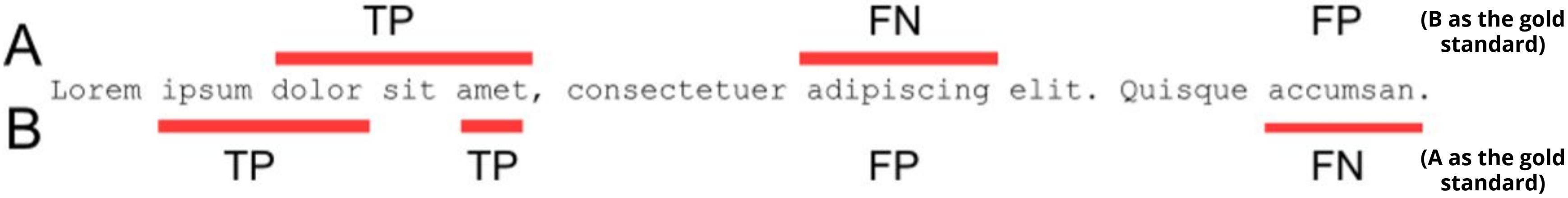


Evaluation

The accuracy in annotation of the relevant parts of text is usually measured by finding *matching* annotations between the *true annotations* in the dataset and the *predicted ones*.

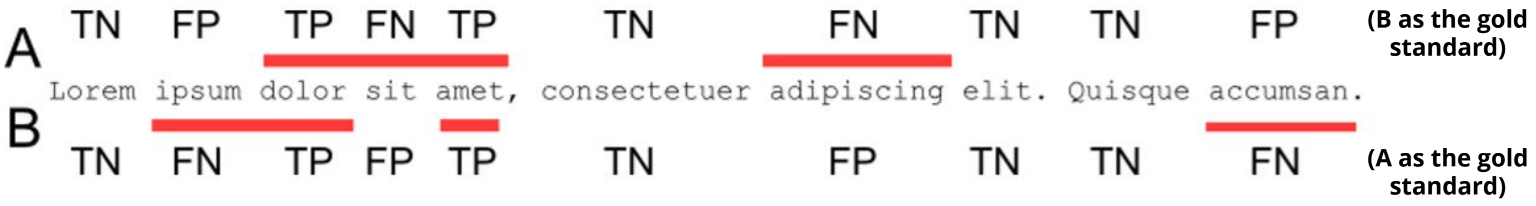
The matching criterium can be **strict** (exact match) or **lenient** (starting at the same word, or just overlapping on some parts).

- Exact match does not capture the gravity of errors.
- Lenient match can be tricked.



Evaluation

Evaluating at the *word level* gives a more graded evaluation, like lenient match, but it cannot be fooled.



Classifying also the separators between words gives an evaluation that coincides with exact match on perfect annotation, yet it is graded.

Wikification

Wikification is the task of linking the relevant parts of a piece of text to the relative Wikipedia entities.

- Identification of relevant parts of text is usually made by matching on a list of *surface forms* for entities.
- When more than one entity is assignable to a piece of text (and also to remove spurious matches) the Wikipedia link graph is exploited.

http://dexter.isti.cnr.it/

http://tagme.di.unipi.it/

Opinion Extraction

Opinion Extraction is focused on establishing *relations* between relevant *entities* of the domain and *subjective expressions* associated to them.

An Opinion Extraction system must be able to perform:

- entity recognition (domain-dependent, possibly including attributes)

  `battery, screen, signal, GPS, memory...`

- subjectivity recognition (polarity can be done in a second time)

  `short battery life, very large screen, not so strong signal...`

Opinion Extraction

Entities can be identified using predefined lists or ontologies.

Once entities are marked, their context is analyzed to find the related subjective expressions.

`The` *`screen`* `is` <u>`very nice`</u>`, but it results in a` <u>`short`</u> *`battery life`*`.`

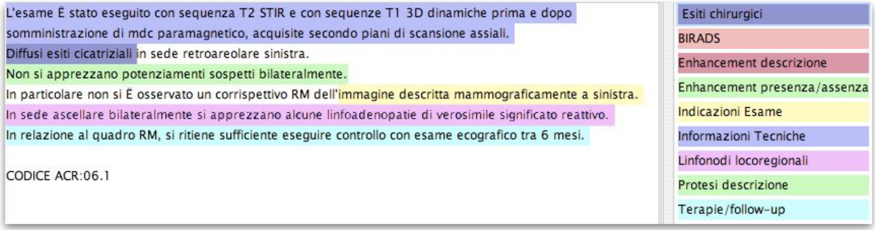
Parsing trees may be of help to connect entities with evaluations, e.g.:

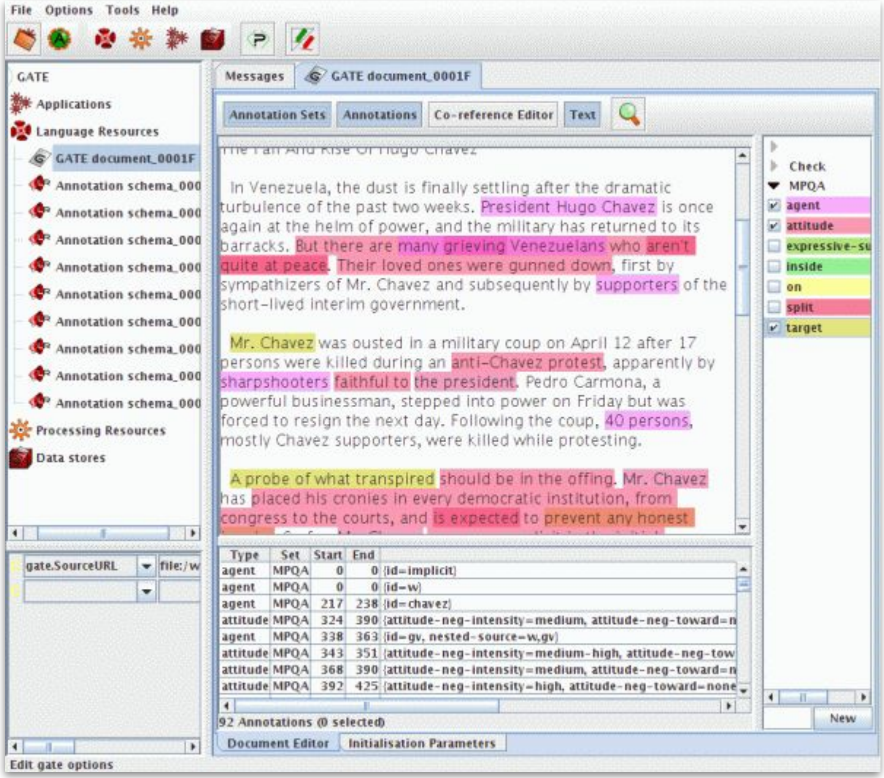
- Verbal phrases may link entity and subjectivity.
- Noun phrases may contains both entity and subjectivity.

(S (S (NP (DT The) (NN *screen*))
        (VP (VBZ is)
        (ADJP (RB <u>very</u>) (JJ <u>nice</u>))))
    (, ,)
    (CC but)
    (S (NP (PRP it))
        (VP (VBZ results)
        (PP (IN in)
        (NP (DT a) (JJ <u>short</u>) (NN *battery*) (NN *life*)))))

GATE

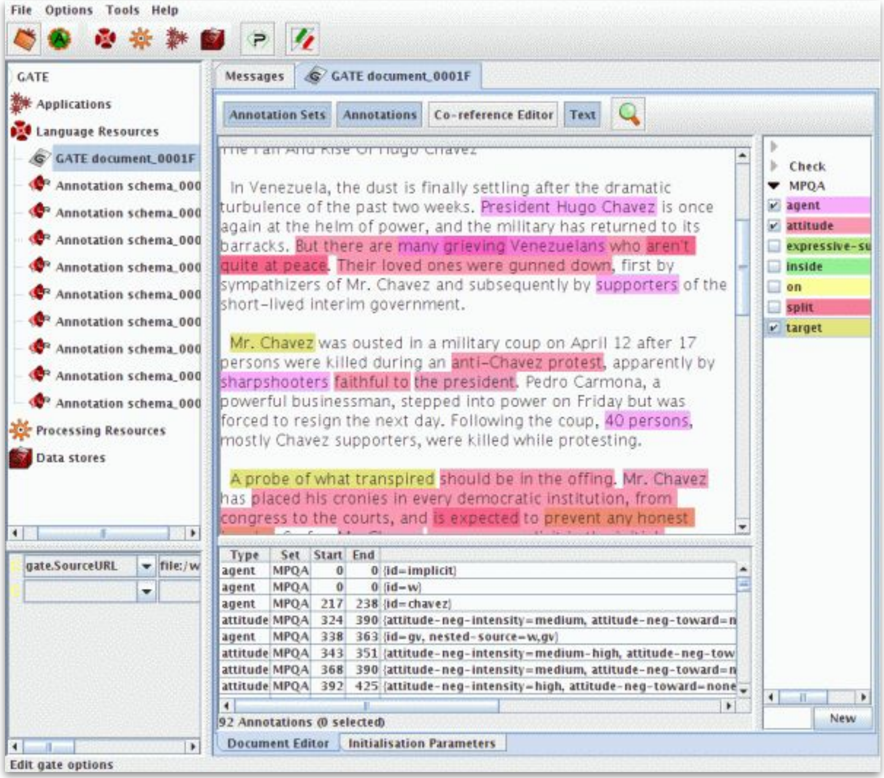
GATE is a text processing tool that includes support for human annotation of entities in text.





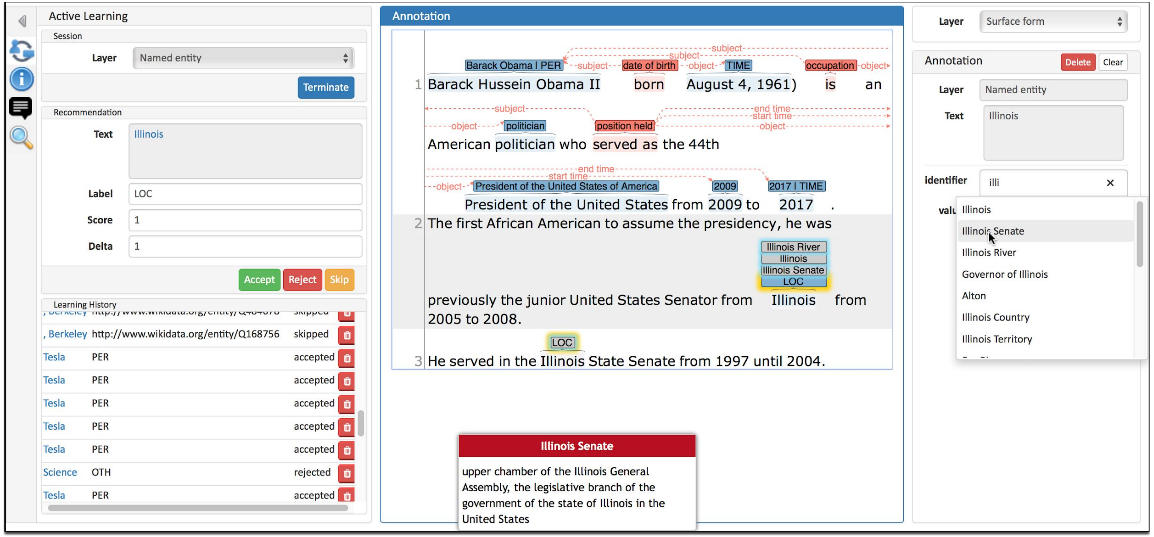
MPQA

The [MPQA Opinion Corpus]() contains news articles from a wide variety of news sources manually annotated for opinions and other private states (i.e., beliefs, emotions, sentiments, speculations, etc.).

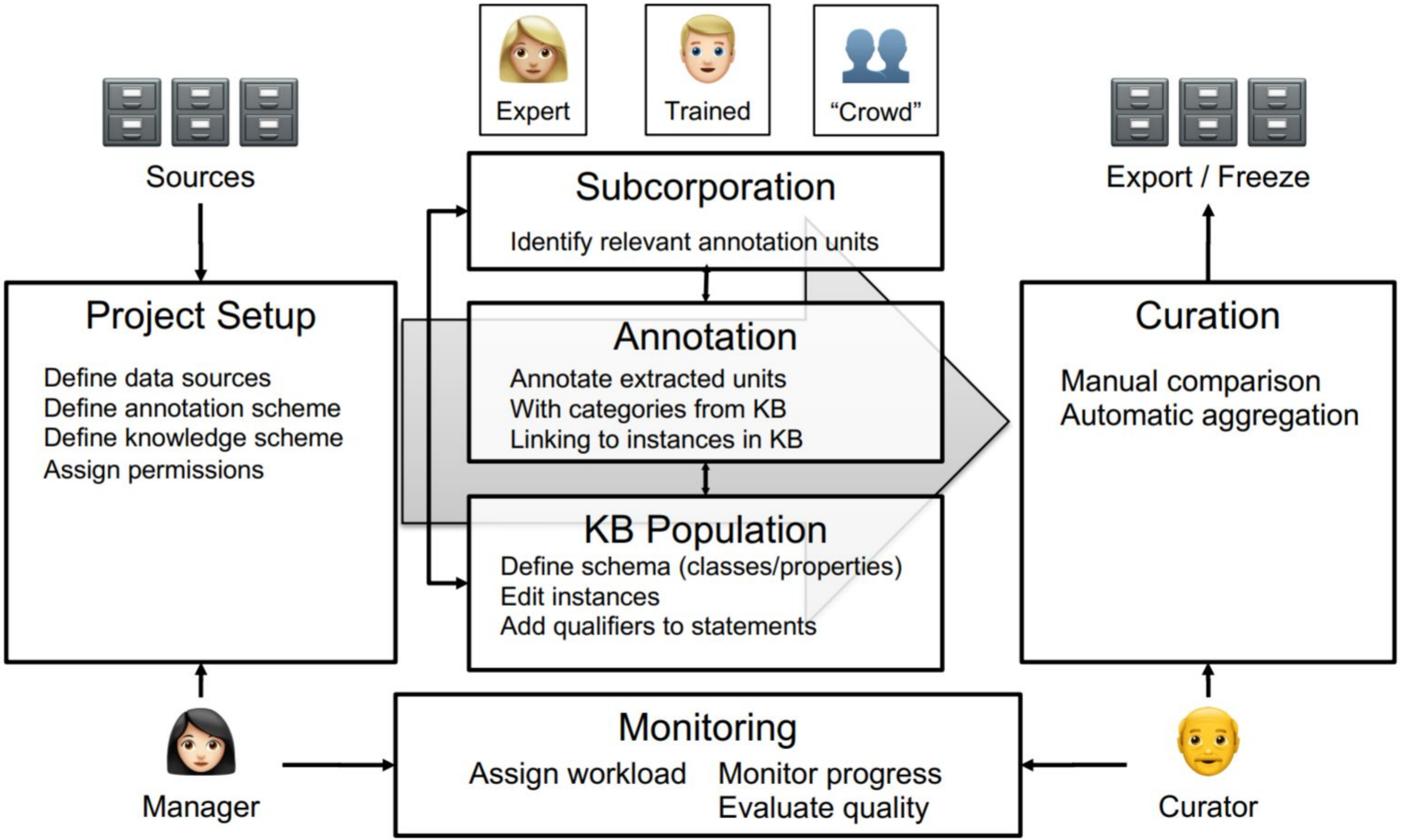


INCEpTION

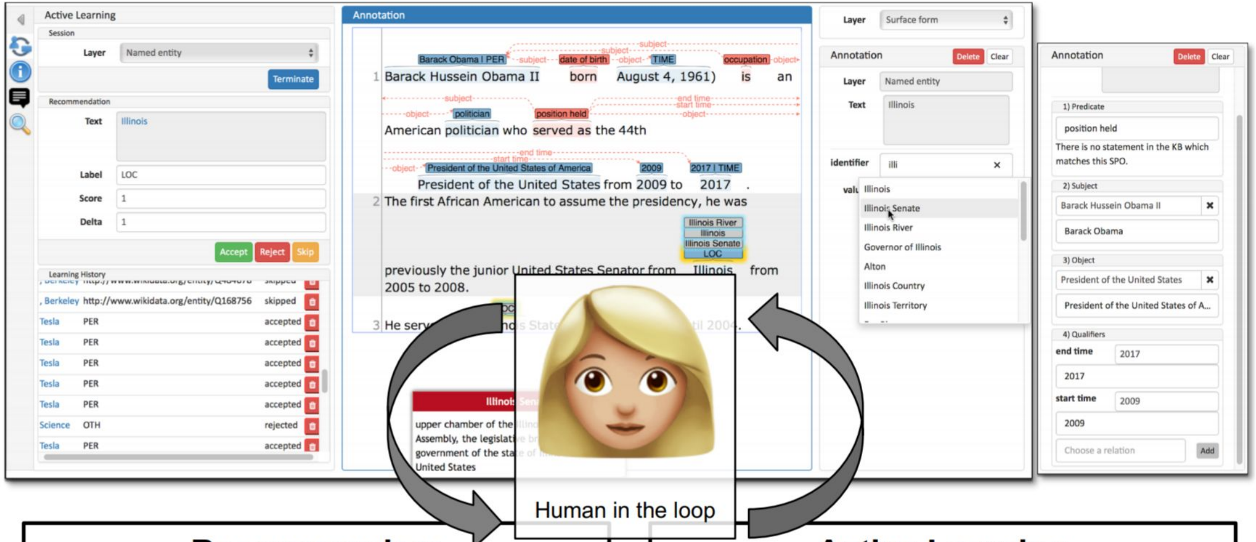
INCEpTION is an annotation platform that supports the definition of custom annotation schema, multi-annotator collaboration (with agreement measurement), and **interactive training of machine learning models for automatic annotation**.



INCEpTION: workflow



INCEpTION: assisted annotation



Recommenders

- Continually learn from the users actions
- Asynchronous training does not slow down user interface
- Automatic evaluation to avoid inaccurate predictions (configurable quality threshold)
- Built-in recommenders: Dictionary-based, OpenNLP-based sequence classifier (part-of-speech, named entities, …)

Active Learning

- Aims at reducing the time to learn by asking specific feedback from the user
- Using uncertainty-sampling strategy
- Compatible with any recommender that provides a confidence score
- User can freely switch between active learning and normal annotation

spaCy

Models from spaCy can be updated with new training data.
New models that annotate user-defined entities can be trained from scratch.

SpaCy uses (simple) custom data structures to represent training data, and supports data conversion from other formats (e.g., BILUO)

```
train_data = [
    ("Uber blew through $1 million a week", [(0, 4, 'ORG')]),
    ("Android Pay expands to Canada", [(0, 11, 'PRODUCT'), (23, 30, 'GPE')]),
    ("Spotify steps up Asia expansion", [(0, 8, "ORG"), (17, 21, "LOC")]),
    ("Google Maps launches location sharing", [(0, 11, "PRODUCT")]),
    ("Google rebrands its business apps", [(0, 6, "ORG")]),
    ("look what i found on google! 😂", [(21, 27, "PRODUCT")])]

doc = Doc(Vocab(), words=["Facebook", "released", "React", "in", "2014"])
gold = GoldParse(doc, entities=["U-ORG", "O", "U-TECHNOLOGY", "O", "U-DATE"])
```

B EGIN	The first token of a multi-token entity.
I N	An inner token of a multi-token entity.
L AST	The final token of a multi-token entity.
U NIT	A single-token entity.
O UT	A non-entity token.

spaCy

```
optimizer = nlp.begin_training(get_data)
for itn in range(100):
    random.shuffle(train_data)
    for raw_text, entity_offsets in train_data:
        doc = nlp.make_doc(raw_text)
        gold = GoldParse(doc, entities=entity_offsets)
        nlp.update([doc], [gold], drop=0.5, sgd=optimizer)
nlp.to_disk("/model")
```