FINET

Context-Aware Fine-Grained Named Entity Typing

Luciano Del Corro*, **Abdalghani Abujabal***,
Rainer Gemulla†, and Gerhard Weikum*

Max-Planck-Institute for Informatics*
University of Mannheim†

Named Entity Typing

The task of detecting ***type(s)*** of named entities in a given ***context*** with respect to a ***type system*** (e.g., WordNet)

“Page plays his guitar on the stage”

guitarist

FINET

A system

- for detecting ***fine-grained types***
- in **short inputs** (e.g., sentences or tweets)
- in a given ***context***
- with respect to ***WordNet***

Context-Aware Typing

“Steinmeier, the German Foreign Minister, ..”

Context-Aware Typing

"Steinmeier, the German Foreign Minister, .."

foreign minister

explicit

Context-Aware Typing

“Steinmeier, the German Foreign Minister, ..”

foreign minister

explicit

“Messi plays soccer”

Context-Aware Typing

"Steinmeier, the German Foreign Minister, .."

foreign minister

explicit

"Messi plays soccer"

soccer player

almost explicit

Context-Aware Typing

"Steinmeier, the German Foreign Minister, .."

foreign minister

explicit

"Messi plays soccer"

soccer player

almost explicit

"Pavano never even made it to the mound"

Context-Aware Typing

“Steinmeier, the German Foreign Minister, ..”

foreign minister

explicit

“Messi plays soccer”

soccer player

almost explicit

“Pavano never even made it to the mound”

baseball player

implicit

Applications

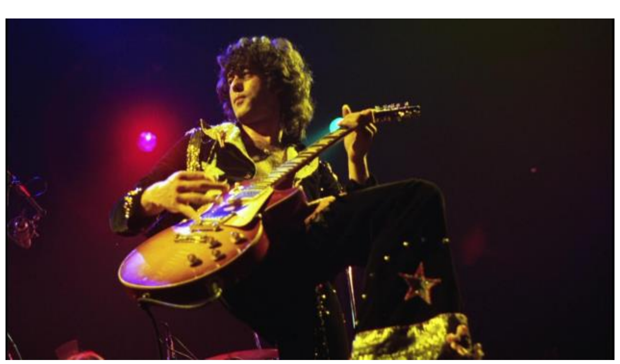
- KB Construction
    - find types for existing entities

Applications

- KB Construction
  - find types for existing entities
- Named Entity Disambiguation
  - “Page played amazingly on the stage”

Musician

Businessman

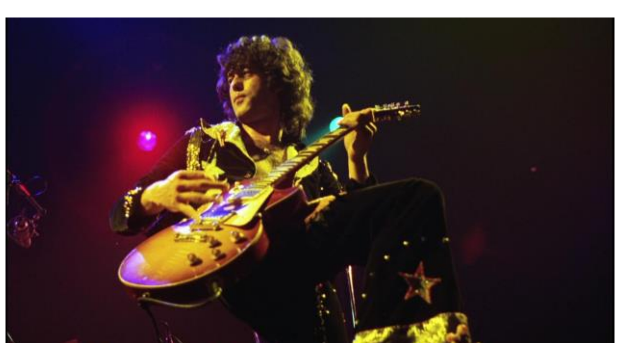


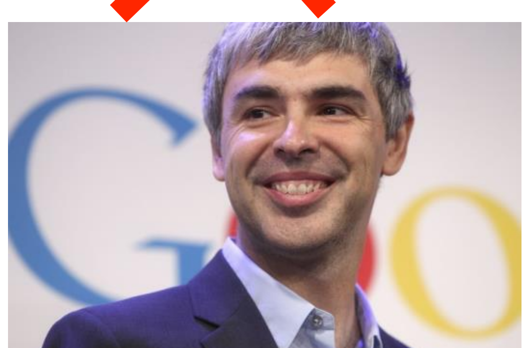
Applications

- KB Construction
  - find types for existing entities
- Named Entity Disambiguation
  - “Page played amazingly on the stage”

Musician

~~Businessman~~





Applications

- KB Construction
  - find types for existing entities
- Named Entity Disambiguation
  - “Page played amazingly on the stage”
- Semantic Search
  - Give me all documents talk about musicians

Supervised Approaches

- Manually labeled data is scarce
  - thousands of types, need sufficient training data for every type

Distantly Supervised Approaches

- Idea: automatically generated data via KB (e.g., Wikipedia)

Distantly Supervised Approaches

- Idea: automatically generated data via KB (e.g., Wikipedia)

"Klitschko is the mayor of Kiev"

"Klitschko is known for his powerful punches"

Distantly Supervised Approaches

- Idea: automatically generated data via KB (e.g., Wikipedia)

boxer
mayor
politician

"Klitschko is the mayor of Kiev"

"Klitschko is known for his powerful punches"

Distantly Supervised Approaches

- Idea: automatically generated data via KB (e.g., Wikipedia)

boxer
mayor
politician

"Klitschko is the mayor of Kiev"

"Klitschko is known for his powerful punches"

Problem: types are context-oblivious

FINET

- Unsupervised
  - Most extractors are unsupervised

FINET

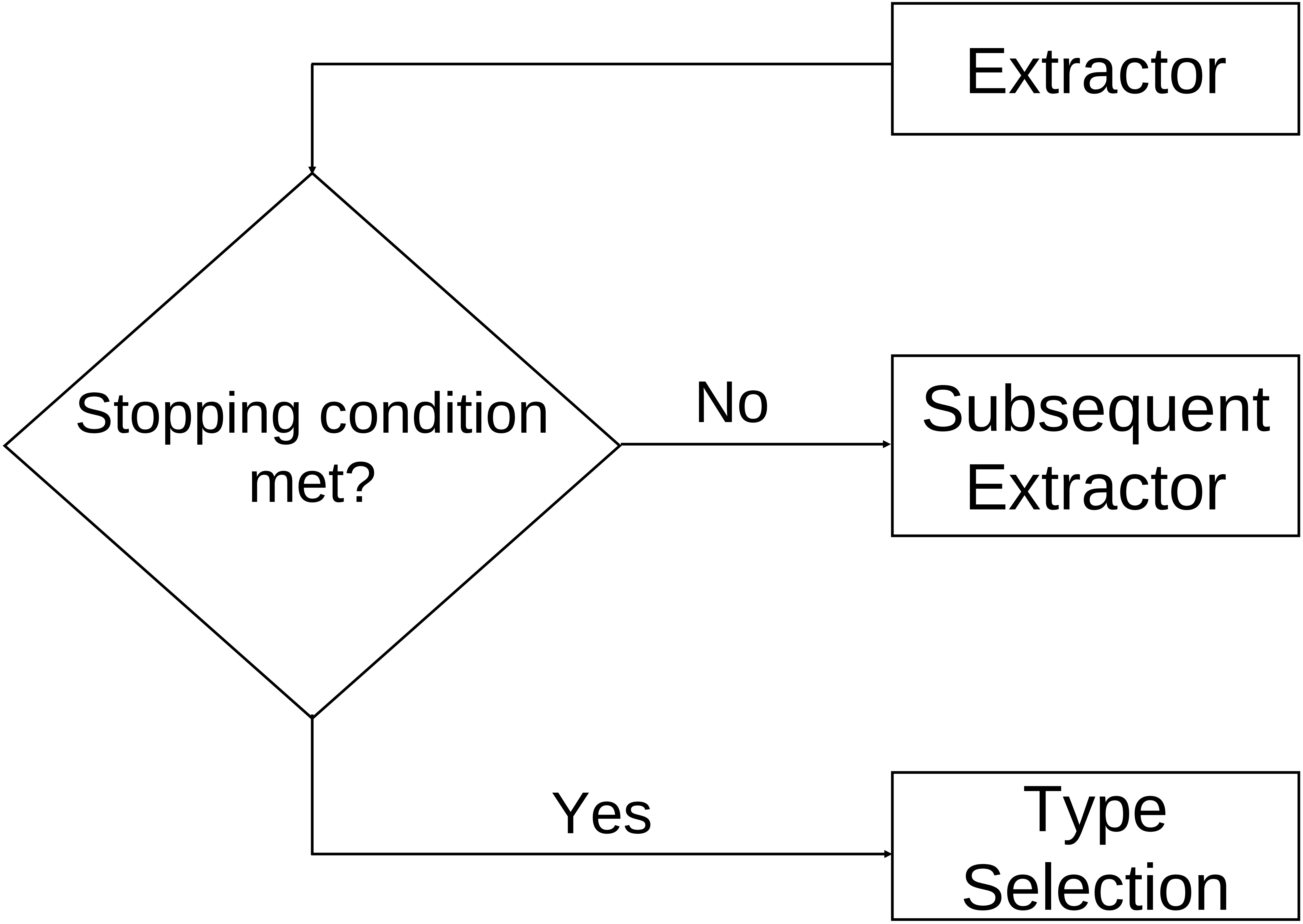
- Unsupervised
  - Most extractors are unsupervised
- Context-aware
  - "Klitschko is the mayor of Kiev" *mayor* *politician*

FINET

- Unsupervised
  - Most extractors are unsupervised
- Context-aware
  - "Klitschko is the mayor of Kiev" *mayor* *politician*
- Super fine-grained
  - WordNet as typing system
    (16K types; per, loc, org)

FINET Overview

1. Preprocessing
2. Candidate Generation
    1. Pattern-based extractor [very explicit]
    2. Mention-based extractor [explicit]
    3. Verb-based extractor [almost explicit]
    4. Corpus-based extractor [implicit]
3. Type Selection (via WSD)


Extractor
Stopping condition met?
No
Subsequent Extractor
Yes
Type Selection

Preprocessing

“Albert Einstein discovered the law of photoelectric effect and he won the Nobel price in 1921”

Preprocessing

"Albert Einstein discovered the law of photoelectric effect and he won the Nobel price in 1921"

- Identify clauses
  - Some extractors operate on clause level (clauses capture local context)

Preprocessing

"Albert Einstein discovered the law of photoelectric effect and he won the Nobel price in 1921"

- Identify coarse-grained types [Stanford NER]
  - FINET restricts its candidates to hyponyms
  - Well studied task: high prec. and recall
  - "Albert Einsten": PER

Preprocessing

"Albert Einstein discovered the law of photoelectric effect and he won the Nobel price in 1921"

- Coreference resolution
  - ("Albert Einstein", "he")

FINET Overview

1. Preprocessing
2. <u>Candidate Generation</u>
    1. Pattern-based extractor [very explicit]
    2. Mention-based extractor [explicit]
    3. Verb-based extractor [almost explicit]
    4. Corpus-based extractor [implicit]
3. Type Selection (via WSD)

Pattern-based Extractor

[final patterns]

targets very explicit types

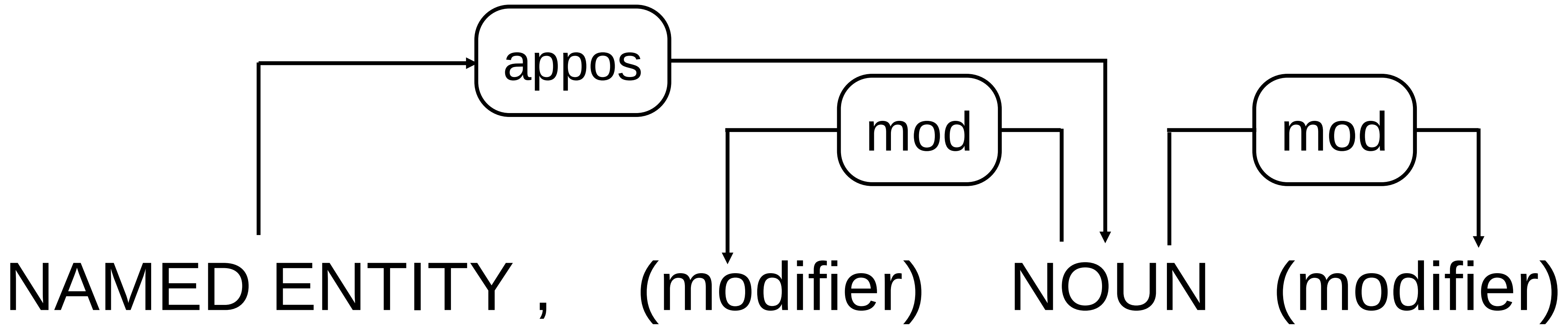
- “Barack Obama, the president of […]”
  - [“Barack Obama”; *president-1*, *president-2*, ..]

Pattern-based Extractor

[final patterns]

targets very explicit types

- “Barack Obama, the president of […]”
  - [“Barack Obama”; *president-1*, *president-2*, ..]

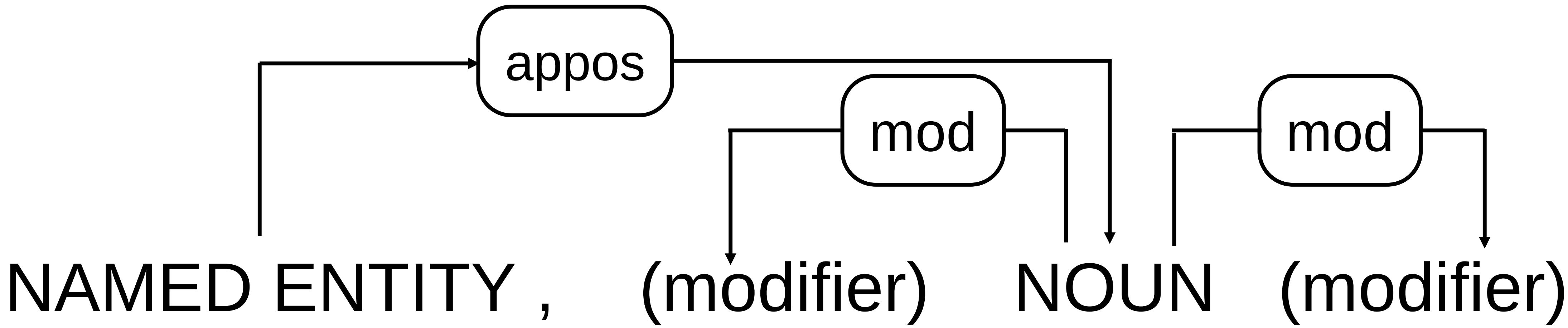


Pattern-based Extractor

[final patterns]

targets very explicit types

- “Barack Obama, the president of […]”
  - [“Barack Obama”; *president-1*, *president-2*, ..]



Stopping Condition: produce at least one type

Pattern-based Extractor
[non-final patterns]

- "Shakespeare's productions"
- production $\xrightarrow{\text{DER}}$ produce $\xrightarrow{\text{DER}}$ producer

["Shakespeare"; *producer-1, producer-2, ..]*

Poss. + transf.

Pattern-based Extractor
[non-final patterns]

- “Shakespeare’s productions”
- production $\xrightarrow{\text{DER}}$ produce $\xrightarrow{\text{DER}}$ producer

["Shakespeare"; *producer-1, producer-2, ..]*

Poss. + transf.

Stopping Condition: KB lookup

Shakespeare	writer-1
Shakespeare	producer-2

Method Overview

1. Preprocessing
2. Candidate Generation
   1. Pattern-based extractor [very explicit]
   2. <u>Mention-based extractor</u> [explicit]
   3. Verb-based extractor [almost explicit]
   4. Corpus-based extractor [implicit]
3. Type Selection (via WSD)

Mention-based Extractor

- “Imperial College London”
  - [“Imperial College London”; *college-1, college-2, ..]*

Mention-based Extractor

- “Imperial College London”
  - [“Imperial College London”; *college-1, college-2, ..]*

Stopping Condition: KB lookup

Method Overview

1. Preprocessing
2. Candidate Generation
    1. Pattern-based extractor [very explicit]
    2. Mention-based extractor [explicit]
    3. Verb-based extractor [almost explicit]
    4. Corpus-based extractor [implicit]
3. Type Selection (via WSD)

Verb-based Extractor

Verb-argument semantic concordance

- Nominalization
  - “play” → “player”

verb deverbal noun

Example 1: Suffixes

- “Messi plays in Barcelona”

Example 1: Suffixes

- “Messi plays in Barcelona”

  play $\xrightarrow{\text{“-er”}}$ player

Example 1: Suffixes

- “Messi plays in Barcelona”

$$\text{play} \xrightarrow{\text{“-er”}} \text{player}$$

play-1 $\xrightarrow{\text{DER}}$ player-1 (*player*)

play-2 $\longrightarrow$ player-2 (*musician*)

play-3 $\longrightarrow$ player-3 (*actor*)

. player-4 (*participant*)

Example 1: Suffixes

- “Messi plays in Barcelona”

play $\xrightarrow{\text{“-er”}}$ player

play-1 $\xrightarrow{\text{DER}}$ player-1 (*player*)

play-2 $\longrightarrow$ player-2 (*musician*)

play-3 $\longrightarrow$ player-3 (*actor*)

. player-4 (*participant*)

[“Messi”; *player, musician, actor, ..]*

Example 1: Suffixes

- "Messi plays in Barcelona"

play $\xrightarrow{\text{"-er"}}$ player

play-1 $\xrightarrow{\text{DER}}$ player-1 (*player*)

play-2 $\longrightarrow$ player-2 (*musician*)

play-3 $\longrightarrow$ player-3 (*actor*)

. player-4 (*participant*)

["Messi"; *player, musician, actor, ..]*

Stopping Condition: KB lookup

Example 2: Synonyms

- “John committed a crime”
  - commit $\xrightarrow{\text{syn}}$ perpetrate $\xrightarrow{\text{DER}}$ perpetrator ["John"; *perpetrator-1]*

Stopping Condition: KB lookup

Method Overview

1. Preprocessing
2. Candidate Generation
    1. Pattern-based extractor [very explicit]
    2. Mention-based extractor [explicit]
    3. Verb-based extractor [almost explicit]
    4. <u>Corpus-based extractor</u> [implicit]
3. Type Selection (via WSD)

Corpus-based Extractor

Distributional hypothesis:
similar entities tend to occur in similar context

- “Messi” & “Cristiano Ronaldo” occur in sport (soccer)
- Key idea: Collect types of similar entities via KB

Word2Vec

- Word vectors represent semantic contexts for a given phrase
- Given a set of phrases, return the *k* most similar phrases with respect to context

“Maradona expects to win in South Africa”

query: {“Maradona”, “South Africa”}

Mention	Type
“Diego Maradona"	<coach-1>, ..
“Parreira"	<coach-1>, ..
“Carlos Alberto Parreira"	<coach-1>, ..
“Dunga"	<coach-1>, ..
…	

“Parreira coached Brazil in South Africa”
“Dunga replaced Parreira after South Africa”

“Maradona expects to win in South Africa”

query: {“Maradona”, “South Africa”}

Mention	Type
“Diego Maradona"	<coach-1>, ..
“Parreira"	<coach-1>, ..
“Carlos Alberto Parreira"	<coach-1>, ..
“Dunga"	<coach-1>, ..
…	

“Parreira coached Brazil in South Africa”
“Dunga replaced Parreira after South Africa”

Stopping Condition: sufficient evidence for types

Method Overview

1. Preprocessing
2. Candidate Generation
    1. Pattern-based extractor [very explicit]
    2. Mention-based extractor [explicit]
    3. Verb-based extractor [almost explicit]
    4. Corpus-based extractor [implicit]
3. <u>Type Selection (via WSD)</u>

Type Selection via Word Sense Disambiguation

- Given an entity and a set of candidate types
  - ["Maradona"; *soccer_player-1*, *football_player-1*, *coach-1*, …]
- Select the best types according to context

Entity Context for WSD

- Entity-oblivious context
  - all words in an input sentence
- Entity-specific context via lexical expansions
  - entity-related words from word vectors

Type Selection via WSD

Naive Bayes trained with word features on WN glosses and labeled data (if available) [ExtendedLesk].

"Maradona expects to win in South Africa"

Entity-oblivious context:

"expects", "win", "South Africa"

Entity-specific context:

"coach", "cup", "striker", "mid-fielder", and "captain"

Experiments

- Datasets
  - 500 random sentences from NYT year 2007
  - 500 random sentences from CoNLL
  - 100 random tweets

Type Granularity

- CG: (artifact, event, person, location, organization)
- FG: ~200 prominent WN types
- SFG: all remaining WN types

System	Type System	Total Types	Top Categories
FINET	WN	16K+	pers, org, loc
HYENA	WN	505	all

System	CG		FG		SFG	
	P	Correct Types	P	Correct Types	P	Correct Types
FINET	87.90	872	72.42	457	70.82	233
FINET (w/o l.)	87.90	872	71.13	436	67.11	204
HYENA	72.40	779	28.26	522	20.65	160

Results on NYT dataset

System	CG		FG		SFG	
	P	Correct Types	P	Correct Types	P	Correct Types
FINET	87.90	872	72.42	457	70.82	233
FINET (w/o l.)	87.90	872	71.13	436	67.11	204
HYENA	72.40	779	28.26	522	20.65	160

Results on NYT dataset

System	CG		FG		SFG	
	P	Correct Types	P	Correct Types	P	Correct Types
FINET	87.90	872	72.42	457	70.82	233
FINET (w/o l.)	87.90	872	71.13	436	67.11	204
HYENA	72.40	779	28.26	522	20.65	160

Results on NYT dataset

Conclusion

- FINET
  - A system for detecting types of named entities
  - Context-aware
  - Unsupervised (mostly)
  - Very fine-grained typing system

Mapping CG types to WN

- persons all descendants of
  - person-1, imaginary, being-1, characterization-3, and operator-2 (10584 in total);
- locations all descendants of
  - location-1, way-1, and landmass-1 (3681 in total);
- organizations all descendants of
  - organization-1 and social group-1 (1968 in total).

Verb-based Extractor

- “Messi plays soccer”
  - “Messi” is a subject
  - “soccer” is direct object
  - Add “soccer” as a noun modifier to the deverbal noun

Verb-based Extractor

- Utilize a corpus of frequent (verb, type) pairs
- “Messi was treated in the hospital”
  - [“Messi”; *patient-1*]

Corpus-based Extractor

- Retrieve 100 most related phrases along with similarity scores

Corpus-based Extractor

- Retrieve 100 most related phrases along with similarity scores
- Filter out non-entity phrases and entities not compatible with CG type

Corpus-based Extractor

- Retrieve 100 most related phrases along with similarity scores
- Filter out non-entity phrases and entities not compatible with CG type
- Traverse the result list until we collect 50% of the total score

Corpus-based Extractor

- Retrieve 100 most related phrases along with similarity scores
- Filter out non-entity phrases and entities not compatible with CG type
- Traverse the result list until we collect 50% of the total score
- If no more that 10 different types were added ➔ add types as candidates