Language Technology
http://cs.lth.se/edan20/
Chapter 10: Partial Parsing

Pierre Nugues

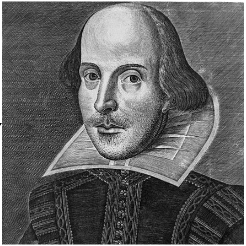
Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

September 23, 2021

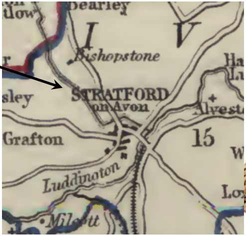
Multiwords

Type	English	French
Prepositions	*to the left hand side*	*À gauche de*
Adverbs	*because of*	*à cause de*
Conjunctions		
Names	*British gas plc.*	*Compagnie générale d'électricité SA*
Titles	*Mr. Smith*	*M. Dupont*
	The President of the United States	*Le président de la République*
Verbs	*give up*	*faire part*
	go off	*rendre visite*

Named Entities: Proper Nouns



William Shakespeare was born and brought up in Stratford-upon-Avon



Others Entities: Common Nouns

Meeting with our guest on the landing at lunchtime

Multiword Annotation

The Message Understanding Conferences (MUC), a benchmarking competition organized by the US military, defined an annotation scheme.

The MUC annotation restricts the annotation to information useful to the funding source: names (named entities), time expressions, and money quantities.

The annotation scheme defines an XML element for three classes: `<ENAMEX>`, `<TIMEX>`, and `<NUMEX>` with which it brackets the relevant phrases in a text.

The phrases can be real multiwords, consisting of two or more words, or restricted to a single word.

<ENAMEX>

The <ENAMEX> element identifies proper nouns and uses a TYPE attribute with three values to categorize them: ORGANIZATION, PERSON, and LOCATION as in

- The <ENAMEX TYPE="PERSON">Clinton</ENAMEX> government
- <ENAMEX TYPE="ORGANIZATION">Bridgestone Sports Co.</ENAMEX>
- <ENAMEX TYPE="ORGANIZATION">European Community</ENAMEX>
- <ENAMEX TYPE="ORGANIZATION">University of California</ENAMEX> in <ENAMEX TYPE="LOCATION">Los Angeles</ENAMEX>

Named Entities

The detection of named entities and multiwords with regular expressions is an extension of word spotting.

Just as for word spotting, we store them in a Python dictionary.

To get a list of names, we can use geographical or name dictionaries, called **gazetteers**.

We can also model patterns, for example for:

```
'<ENAMEX> M. Dupont </ENAMEX>'
```

and

```
'<NUMEX> 200 euros </NUMEX>'
```

ELIZA: Word Spotting and Template Matching

User	**Psychotherapist**
. . . I like X. . .	*Why do you like X?*
. . . I am X. . .	*How long have you been X?*
. . . father. . .	*Tell me more about your father*

Word Spotting with Regular Expressions

Models for the dialogue pairs using a set of regular expressions:

```
dialogue_pairs = {
    '.*I am not (.+)': [
        r'Why aren\'t you \1'
    ],
    '.*I am (.+)': [
        r'How long have you been \1'
    ],
    '.*I like (.+)': [
        r'Why do you like \1'
    ],
    '.*I remember (.+)': [
        r'Do you often think of \1',
        'What else do you remember',
        r'Why do you remember \1 just now'
    ],
    '.*(father|mother|brother|sister).*': [
        r'Please tell me more about your \1'
```

Word Spotting with Regular Expressions (Continued)

```
dialogue_pairs = {
    ...
    '^no$': [
        'Why are you so negative?',
        'Why not?'
    ],
    '^\p{L}+$': [
        'Tell me more...'
    ],
    # When nothing matches
    '.*': [
        'I am not sure I understand you fully',
        'Please go on',
        'What does that suggest to you'
    ]
}
```

A simplified version of ELIZA in Python

```
def match_utterance(user_input):
    """
    match_utterance accepts the user's utterance and
    tries to find a template matching it
    """
    for key in dialogue_pairs:
        matches = list(re.finditer(key, user_input))
        if matches:
            choice = np.random.randint(len(dialogue_pairs[key]
            answer = re.sub(key, dialogue_pairs[key][choice],
                            user_input, count=1)
            return answer
```

ELIZA in Python

```
# The main loop reads the input and calls match_utterance
# It stops when the input is the word bye.
print('Hello, I am ELIZA. How can I help you?')
user_input = input('User: ')
while user_input.lower() != 'bye':
    answer = match_utterance(user_input)
    print('Eliza:', answer)
    user_input = input('User: ')
print('Eliza:', 'Bye')
```

A Dialogue with ELIZA

Hello, I am ELIZA. How can I help you?
User: yes
Eliza: Tell me more...
User: I like apples
Eliza: Why do you like apples
User: I went out with my father
Eliza: Please tell me more about your father
User: no
Eliza: Why are you so negative?
User: I am cold
Eliza: How long have you been cold
User: bye
Eliza: Bye

Regular Expressions for Named Entities

```
ne_pairs = {
    'in front of': [
        'in_front_of'
    ],
    'in front': [
        'in_front'
    ],
    'give up': [
        'give_up'
    ],
    'M\. (\p{Lu}\p{L}+)': [
        r'<ENAMEX> M. \1 </ENAMEX>'
    ],
    r'(\p{N}+) euros': [
        r'<NUMEX> \1 euros </NUMEX>'
    ]
}
```

Longest Match

The regex

```
'in front'|'in front of'
```

does not work on:

The car in front of the house

with the code

```
input = 'The car in front of the house'
re.search('in front|in front of', input)
<regex.Match object; span=(8, 16), match='in front'>
```

We need to have the regexes in the right order:

```
'in front of': [
    'in_front_of'
],
'in front': [
    'in_front'
]
```

Noun Groups

English	French	German
The waiter *is bringing* **the very big dish** *on* **the table**	**Le serveur** *apporte* **le très grand plat** *sur* **la table**	**Der Ober** *bringt* **die sehr große Speise** *an* **den Tisch**
Charlotte *has eaten* **the meal** *of* **the day**	**Charlotte** *a mangé* **le plat** *du* **jour**	**Charlotte** *hat* **die Tagesspeise** *gegessen*

Verb Groups

English	French	German
The waiter **is bringing** *the very big dish on the table*	*Le serveur* **apporte** *le très grand plat sur la table*	*Der Ober* **bringt** *die sehr große Speise an den Tisch*
Charlotte **has eaten** *the meal of the day*	*Charlotte* **a mangé** *le plat du jour*	*Charlotte* **hat** *die Tagesspeise* **gegessen**

Noun Groups

```
nominal([NOUN | NOM]) --> noun(NOUN), nominal(NOM).
nominal([N]) --> noun(N).

noun(N) --> common_noun(N).
noun(N) --> proper_noun(N).

noun_group([PRO]) --> pronoun(PRO).
noun_group([D | N]) --> det(D), nominal(N).
noun_group(N) --> nominal(N).
```

Tagging Techniques to Extract Groups

Group detection – chunking – can be reframed as a tagging operation.

From: [$_{NG}$ The government $_{NG}$] has [$_{NG}$ other agencies and instruments $_{NG}$] for pursuing [$_{NG}$ these other objectives $_{NG}$] .

To: *The*/I *government*/I *has*/O *other*/I *agencies*/I *and*/I *instruments*/I *for*/O *pursuing*/O *these*/I *other*/I *objectives*/I ./O

From: Even [$_{NG}$ Mao Tse-tung $_{NG}$] [$_{NG}$'s China $_{NG}$] began in [$_{NG}$ 1949 $_{NG}$] with [$_{NG}$ a partnership $_{NG}$] between [$_{NG}$ the communists $_{NG}$] and [$_{NG}$ a number $_{NG}$] of [$_{NG}$ smaller, non-communists parties $_{NG}$] .

To: *Even*/O *Mao*/I *Tse-tung*/I *'s*/B *China*/I *began*/O *in*/O *1949*/I *with*/O *a*/I *partnership*/I *between*/O *the*/I *communists*/I *and*/O *a*/I *number*/I *of*/O *smaller*/I *,*/I *non-communists*/I *parties*/I ./O

Other Chunking Schemes

Tjong and Venstra (1999) created 3 other schemes: IOB1, IOB2, IOE1, and IOB2. A 5th tagset, BIOES, is gaining popularity:

- IOB1 : Inside, Outside, Between
- IOB2 : Begin, Inside, Outside, possibly the most popular
- IOE1 : Inside, Outside, End (between two chunks)
- IOE2 : Inside, Outside, End
- BIOES : Begin, Inside, Outside, End, and Singleton, the most efficient one.

Other Chunking Schemes

IOB1	Even/O Mao/I Tse-tung/I 's/B China/I began/O in/O 1949/I with/O a/I partnership/I between/O the/I communists/I and/O a/I number/I of/O smaller/I, non-communists/I parties/I
IOB2	Even/O Mao/B Tse-tung/I 's/B China/I began/O in/O 1949/B with/O a/B partnership/I between/O the/B communists/I and/O a/B number/I of/O smaller/B, non-communists/I parties/I
IOE1	Even/O Mao/I Tse-tung/E 's/I China/I began/O in/O 1949/I with/O a/I partnership/I between/O the/I communists/I and/O a/I number/I of/O smaller/I, non-communists/I parties/I
BIOES	Even/O Mao/B Tse-tung/E 's/B China/E began/O in/O 1949/S with/O a/B partnership/E between/O the/B communists/E and/O a/B number/E of/O smaller/B, non-communists/I parties/E

Multiple Categories of Chunks

Extendable to any type of chunks: nominal, verbal, etc.
For the IOB scheme, this means tags such as I.Type, O.Type, and B.Type, Types being NG, VG, PG, etc.
In CoNLL 2000, ten types of chunks

Word	POS	Group	Word	POS	Group
He	PRP	B-NP	*to*	TO	B-PP
reckons	VBZ	B-VP	*only*	RB	B-NP
the	DT	B-NP	*£*	#	I-NP
current	JJ	I-NP	*1.8*	CD	I-NP
account	NN	I-NP	*billion*	CD	I-NP
deficit	NN	I-NP	*in*	IN	B-PP
will	MD	B-VP	*September*	NNP	B-NP
narrow	VB	I-VP	*.*	.	O

Noun groups (NP) are in red and verb groups (VP) are in blue.

Evaluation: Accuracy, precision, and recall

For noun groups with the predicted output:

Word	POS	Group		Predicted	Word	POS	Group		Predicted
He	PRP	B-NP		B-NP	*to*	TO	B-PP		B-PP
reckons	VBZ	B-VP		B-VP	*only*	RB	B-NP	X	B-NP
the	DT	B-NP	X	B-NP	*£*	#	I-NP	X	I-NP
current	JJ	I-NP	X	**B-NP**	*1.8*	CD	I-NP	X	**B-NP**
account	NN	I-NP	X	I-NP	*billion*	CD	I-NP	X	I-NP
deficit	NN	I-NP	X	I-NP	*in*	IN	B-PP		B-PP
will	MD	B-VP		B-VP	*September*	NNP	B-NP		B-NP
narrow	VB	I-VP		I-VP	*.*	.	O		O

There are 16 chunk tags, 14 are correct: Accuracy $= \frac{14}{16} = 0.875$
There are 4 noun groups, the system retrieved 2 of them: Recall $= \frac{2}{4} = 0.5$
The system identified 6 noun groups, two are correct: Precision $= \frac{2}{6} = 0.33$
Harmonic mean $= 2 \times \frac{0.33 \times 0.5}{0.33 + 0.5} = 0.4$

IOB Annotation for Named Entities

CoNLL 2002

Words	Named entities
Wolff	B-PER
,	O
currently	O
a	O
journalist	O
in	O
Argentina	B-LOC
,	O
played	O
with	O
Del	B-PER
Bosque	I-PER
in	O
the	O
final	O
years	O
of	O
the	O
seventies	O
in	O
Real	B-ORG
Madrid	I-ORG
.	O

CoNLL 2003

Words	POS	Groups	Named entities
U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Chunking Algorithms

We can apply statistical and symbolic methods to chunking:

1. Brill's method with templates adapted to groups.
2. Stochastic and machine learning methods similar to POS tagging: logistic regression, support vector machines, or decision trees

Feature Engineering (I)

CoNLL 2000 baseline: Use t_i to predict $chunk\_tag_i$

He	*reckons*	*the*	*current*	*account*	*deficit*	*will*	*narrow*
PRP	VBZ	DT	JJ	NN	NN	MD	VB
B-NP	B-VP	B-NP	I-NP	I-NP	I-NP	B-VP	I-VP

to	*only*	*#*	*1.8*	*billion*	*in*	*September*	*.*
TO	RB	#	CD	CD	IN	NNP	.
B-PP	B-NP	I-NP	I-NP	I-NP	B-PP	B-NP	O

F-measure: 77.07

Feature Engineering (II)

Second experiment using decision trees: Use t_{i-1}, t_i to predict $chunk\_tag_i$

He	*reckons*	*the*	*current*	*account*	*deficit*	*will*	*narrow*
PRP	VBZ	DT	JJ	NN	NN	MD	VB
B-NP	B-VP	B-NP	I-NP	I-NP	I-NP	B-VP	I-VP

to	*only*	*#*	*1.8*	*billion*	*in*	*September*	*.*
TO	RB	#	CD	CD	IN	NNP	.
B-PP	B-NP	I-NP	I-NP	I-NP	B-PP	B-NP	O

F-measure: 81.88

Feature Engineering (III)

Third experiment: Use t_{i-2}, t_{i-1}, t_i to predict $chunk\_tag_i$

He	*reckons*	*the*	*current*	*account*	*deficit*	*will*	*narrow*
PRP	VBZ	DT	JJ	NN	NN	MD	VB
B-NP	B-VP	B-NP	I-NP	I-NP	I-NP	B-VP	I-VP

to	*only*	*#*	*1.8*	*billion*	*in*	*September*	*.*
TO	RB	#	CD	CD	IN	NNP	.
B-PP	B-NP	I-NP	I-NP	I-NP	B-PP	B-NP	O

F-measure: 82.84

Dynamic Features

So far, we used "static" features extracted from a first annotation, for example, the words and their part of speech: $w_{i-1}, t_{i-1}, w_i, t_i$

We can add dynamic features that will reuse the value of the preceding (and just obtained) chunk tag.

It is possible to reuse chunk tags to the left in case of left-to-right parsing and to the right in case of right-to-left parsing

This corresponds to recurrent neural networks

Feature Engineering (IV)

Fourth experiment: Use $w_i, t_{i-1}, t_i, t_{i+1}, chunk\_tag_{i-1}$ to predict $chunk\_tag_i$. All words with a frequency less than ∼100 mapped onto a unique symbol (`RARE_WORD`).

He	*reckons*	*the*	*current*	*account*	*deficit*	*will*	*narrow*
PRP	VBZ	DT	JJ	NN	NN	MD	VB
B-NP	B-VP	B-NP	I-NP	I-NP	I-NP	B-VP	I-VP

to	*only*	*#*	*1.8*	*billion*	*in*	*September*	*.*
TO	RB	#	CD	CD	IN	NNP	.
B-PP	B-NP	I-NP	I-NP	I-NP	B-PP	B-NP	O

F-measure: 90.17

Kudoh and Matsumoto (2000)

Kudoh and Matsumoto (2000) won the CoNLL-2000 shared task.
They used static and dynamic features in the Yamcha system
Typically, a feature vector consists of 10 static parameters:
$w_{i-2}, t_{i-2}, w_{i-1}, t_{i-1}, w_i, t_i, w_{i+1}, t_{i+1}, w_{i+2}, t_{i+2}$
And two dynamic parameters: $chunk\_tag_{i-2}, chunk\_tag_{i-1}$
Kudoh and Matsumoto (2000) experimented various feature vectors, forward and backward parsing, as well as the four annotation schemes.
Their classifiers used support vector machines.

Example from Kudoh and Matsumoto (2000)

Three lines or columns representing the words, the parts of speech, and the groups.

He	*reckons*	*the*	*current*	*account*	*deficit*	*will*	*narrow*
PRP	VBZ	DT	JJ	NN	NN	MD	VB
B-NP	B-VP	B-NP	I-NP	I-NP	I-NP	B-VP	I-VP

to	*only*	*#*	*1.8*	*billion*	*in*	*September*	*.*
TO	RB	#	CD	CD	IN	NNP	.
B-PP	B-NP	I-NP	I-NP	I-NP	B-PP	B-NP	O

Example from Kudoh and Matsumoto (2000)

Words	POS	Groups	
BOS	BOS	BOS	**Padding**
BOS	BOS	BOS	
He	PRP	B-NP	
reckons	VBZ	B-VP	
the	DT	B-NP	
current	JJ	I-NP	
account	NN	I-NP	
deficit	NN	I-NP	**Input features**
will	MD	B-VP	
narrow	VB	I-VP	**Predicted tag**
to	TO	B-PP	↓
only	RB	B-NP	
£	#	I-NP	
1.8	CD	I-NP	
billion	CD	I-NP	
in	IN	B-PP	
September	NNP	B-NP	
.	.	O	
EOS	EOS	EOS	**Padding**
EOS	EOS	EOS	

Message Understanding Conferences

The Message Understanding Conferences (MUCs) measure the performance of information extraction systems.
They are competitions organized by an agency of the US department of defense, the DARPA
The competitions have been held regularly until MUC-7 in 1997.
The performances improved dramatically in the beginning and stabilized then.
MUCs are divided into a set of tasks that have been changing over time.
The most basic task is to extract people and company names.
The most challenging one is referred to as information extraction.

Information Extraction

Information extraction consists of:

- The analysis of pieces of text ranging from one to two pages,
- The identification of entities or events of a specified type,
- The filling of a pre-defined template with relevant information from the text.

Information extraction then transforms free texts into tabulated information.

An Example

San Salvador, 19 Apr 89 (ACAN-EFE) – [TEXT] Salvadoran President-elect Alfredo Cristiani condemned the terrorist killing of Attorney General Roberto Garcia Alvarado and accused the Farabundo Marti National Liberation Front (FMLN) of the crime...

Garcia Alvarado, 56, was killed when a bomb placed by urban guerrillas on his vehicle exploded as it came to a halt at an intersection in downtown San Salvador...

Vice President-elect Francisco Merino said that when the attorney general's car stopped at a light on a street in downtown San Salvador, an individual placed a bomb on the roof of the armored vehicle...

According to the police and Garcia Alvarado's driver, who escaped unscathed, the attorney general was traveling with two bodyguards. One of them was injured.

The Template

Template slots	**Information extracted from the text**
Incident: Date	19 Apr 89
Incident: Location	El Salvador: San Salvador (city)
Incident: Type	Bombing
Perpetrator: Individual ID	*urban guerrillas*
Perpetrator: Organization ID	*FMLN*
Perpetrator: Organization confidence	Suspected or accused by authorities: *FML*
Physical target: Description	*vehicle*
Physical target: Effect	Some damage: *vehicle*
Human target: Name	*Roberto Garcia Alvarado*
Human target: Description	*Attorney general*: *Roberto Garcia Alvarac* *driver* *bodyguards*
Human target: Effect	Death: *Roberto Garcia Alvarado* No injury: *driver* Injury: *bodyguards*

FASTUS

The FASTUS system has been designed at the Stanford Research Institute to extract information from free-running text

FASTUS uses partial parsers that are organized as a cascade of finite-state automata.

It includes a tokenizer, a multiword detector, and a group detector as first layers.

Verb groups are tagged with active, passive, gerund, and infinitive features.

Then FASTUS combines some groups into more complex phrases and uses extraction patterns to fill the template slots.

FASTUS' Architecture

Sentence

Tokenizer

Multiwords

Part-of-speech tagging

Group detection (or chunking)

Evaluation

The Message Understanding Conferences have introduced a metric to evaluate the performance of information extraction systems using three figures.
They are borrowed them from library science

	Relevant documents	**Irrelevant documents**
Retrieved	A	B
Not retrieved	C	D

Recall, Precision, and the F-Measure

Recall measures how much relevant information the system has retrieved.

$$\text{Recall} = \frac{A}{A \cup C}.$$

Precision is the accuracy of what has been returned

$$\text{Precision} = \frac{A}{A \cup B}.$$

Recall and precision are combined into the **F-measure**, which is defined as the harmonic mean of both numbers:

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}.$$