Named Entity Recognition
rule-based method

Fang Li
Dept. of Computer Science & Technology
Contents

• Regular Expression
• Problems on Entity Identification
• Knowledge Engineering Approach
Regular expressions

- A formal language for specifying text strings:
  - A kind of pattern
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks
Regular expressions

**Metacharacters**: Twelve characters

- the backslash \,
- the caret ^,
- the dollar sign $,
- the period or dot .,
- the vertical bar or pipe symbol |,
- the question mark ?,
- the asterisk or star *,
- the plus sign +,
- the opening parenthesis (,
- the closing parenthesis ),
- the opening square bracket [,  
- the opening curly brace {.

Shorthand for character class:

- \d: a digit.
- \w: a "word character" (alphanumeric characters plus underscore).
- \s: matches a whitespace character (includes tabs and line breaks).
Regular Expressions: Disjunctions

### Letters inside square brackets []

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[wW]oodchuck</td>
<td>Woodchuck, woodchuck</td>
</tr>
<tr>
<td>[1234567890]</td>
<td>Any digit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A-Z]</td>
<td>An upper case letter</td>
</tr>
<tr>
<td>[a-z]</td>
<td>A lower case letter</td>
</tr>
<tr>
<td>[0-9]</td>
<td>A single digit</td>
</tr>
<tr>
<td></td>
<td>Chapter 1: Down the Rabbit Hole</td>
</tr>
</tbody>
</table>

2014/10/11 lecture of Internet-based IE Technology
## Regular Expressions: Negation in Disjunction

### Negations \[^Ss\]

- **Carat after the opening square bracket** negates the character class.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[^A-Z]</td>
<td>Not an upper case letter</td>
</tr>
<tr>
<td>[^Ss]</td>
<td>Neither ‘S’ nor ‘s’</td>
</tr>
<tr>
<td>q[^e]</td>
<td>Not e</td>
</tr>
<tr>
<td>a^b</td>
<td>The pattern a carat b</td>
</tr>
</tbody>
</table>
More Disjunction

The pipe | for disjunction

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>yours</td>
<td>mine</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>(cat</td>
<td>dog) food</td>
</tr>
</tbody>
</table>

**Repetition:** Use **curly braces** to specify a specific amount of repetition. Examples:

\b[1-9][0-9]{3}\b  match a number between 1000 and 9999.
\b[1-9][0-9]{2,4}\b  matches a number between 100 and 99999.
## Regular Expressions: \(? \ast \ + \ .\)

Makes the preceding token optional

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>colou?r</code></td>
<td>Optional previous char</td>
</tr>
<tr>
<td></td>
<td><code>color</code> <code>colour</code></td>
</tr>
<tr>
<td><code>oo*h!</code></td>
<td>0 or more of previous char</td>
</tr>
<tr>
<td></td>
<td><code>oh! ooh! oooh! ooooh!</code></td>
</tr>
<tr>
<td><code>o+h!</code></td>
<td>1 or more of previous char</td>
</tr>
<tr>
<td></td>
<td><code>oh! ooh! oooh! ooooh!</code></td>
</tr>
<tr>
<td><code>baa+</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>baa baaa baaaa baaaaaa</code></td>
</tr>
<tr>
<td><code>beg.n</code></td>
<td>a single character, except line break characters</td>
</tr>
</tbody>
</table>
|           | `begin begun begun beg3n`
Anchors: \^ \$ Anchors do not match any characters. They **match a position**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>^[A-Z]</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>^[^A-Za-z]</td>
<td>1 “Hello”</td>
</tr>
<tr>
<td>.$</td>
<td>The end.</td>
</tr>
<tr>
<td>.$</td>
<td>The end? The end!</td>
</tr>
<tr>
<td>\b</td>
<td>matches at a word boundary.</td>
</tr>
</tbody>
</table>
Example

- Find me all instances of the word “the” in a text.

  - misses capitalized examples
  - [tT]he incorrectly returns other or theology

[^a-zA-Z][tT]he[^a-zA-Z]

How to describe the regular expression of an email address?
Summarization for regular expression

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
- Can be very useful in capturing generalizations
``TWA has not been a normal company,'' said Robert Peiser, chief financial officer.

We can not substantiate the claims.

Entities include:

- **Named entities**: TWA, Robert Peiser ...
- **pronoun entities**: we, ...
- **nominal entities**: the company, ...

**Named entities** are the most important one among the 3 categories which is the anchor point for IE.
Difficulties of NE recognition

- Potential set of NE is too large to include in dictionaries/Gazetteers.
- Names changing constantly.
- Names appear in many variant forms. E.g. John Smith, Mr Smith or John
- Subsequent occurrences of names might be abbreviated.
- Ambiguity of NE types. E.g. John Smith is a person name or a company name? depends on:
  - Internal structure: Mr. John Smith
  - Context: The new company, John Smith will make….
  - Ambiguity: Washington is a person or a location?
Basic Steps for Named Entity Recognition

- Build linguistic patterns or rules to identify Entities or Relations

“Dr. Yiming Yang was appointed as CEO of IBM at ...”

“Smith was appointed as chairman of the account board.

→

**Pattern:** person **was appointed as** post of company
Basic Steps for Named Entity Recognition (cont.)

- Apply rules or patterns to text and extraction

"Smith was appointed as Akim of Akmola region” →

**Person:** Smith

**Post:** Akim (head of local government)

**Company:** Akmola region

**IBM named smith as president** → ???

2014/10/11 lecture of Internet-based IE Technology
Pattern Needs

- general enough to have a broad applicability, but specific enough to be consistently reliable over a large number of texts.

- Too general

Person, post convinced company ...

- Too specific

company named person to post
Who build patterns?

- Pattern recognition aims at classifying data (patterns) based on either a priori knowledge that is acquired by human experts or on knowledge automatically learned from data.
Difficulties to Collect the Patterns

- lack of sufficient training examples.
- **Natural language is very varied.** capturing all possible variations in the examples and having sufficient overlap in the examples to discriminate good patterns from noisy patterns is almost impossible.
- **features is very large**, but only few of them are active in each example, and only a small fraction of them are relevant to the target concept.
- **Ambiguous features**, the individual features and their values are often ambiguous markers of several classes;
- **extraction algorithms** need to perform fast computations and their computational complexity should be taken an eye on.
Difficulties to Collect the Patterns (cont.)

For example:

- Different words:
  named, appointed, selected, chosen, promoted, ...

- Different constructions:
  IBM named Fred president
  IBM announced the appointment of Fred as president
  Fred, who was named president by IBM

- Different names:
  George H. W. Bush, former President Bush, 41
Difficulties to collect the patterns (cont.)

- **Ambiguity**
  
  Fred’s appointment as professor vs. Fred’s 3 PM appointment with the dean
  
  outbreak of typhoid (伤寒) vs. outbreak of violence (暴力)

- **Complex structures**
  
  For the Federal Election Commission, Bush picked Justice Department employee and former Fulton County, Ga., Republican chairman **Hans von Spakovsky** for one of three openings.

- **Reference**
  
  - **George Garrick** has served as president of Sony USA for 13 years. *The company* announced his retirement effective next May.
  
  - **IBM** announced several new appointments yesterday. Fred Smith was named **head of research**.
Features

According to its position in the text:

- Features that occur in the information unit itself, such as the composition of letters and digits of an entity name.
- Features that close neighborhood or context window of the token string to be classified.
- Features that occur in the complete document or document collection.
Features (cont.)

Features according to their types

- **Lexical**: variations concerning punctuation (USA versus U.S.A), capitalization (e.g., Citibank versus CITIBANK)

- **Syntactic**: The part-of-speech of a word.

- **Semantic**: refer to semantic classifications of single- or multi-word information units.

- **Discourse features**: refer to features the values of which are computed by using text fragments, i.e., a discourse or a connected speech or writing, larger than the sentence.
Rule-based methods

- Use a lexicon to identify some named entities.
- Identify possible parts of names with lexical features
- Write rules to recognize names
  - Take advantage of capitalization
  - Take advantage of internal structure
  - Mumble Mumble City → probably a location
  - Mumble Mumble GmbH → probably a company
- Run over a corpus, find errors:
  - General Electric is a company, not a general
  - Yesterday IBM Corp. announced ...
- A large set of complex rules will be the result
Use a lexicon to identify some named entities

- Advantages - Simple, fast, language independent, easy to retarget.
- Disadvantages - collection and maintenance of lists, cannot deal with name variants, cannot resolve ambiguity and include all.
- How to find a lexicon?
Using Gazetteer (Lexicon)

- Online phone directories and yellow pages for person and organization names
  - Locations lists:
    US GEOnet Names Server (GNS) data – 3.9 million locations with 5.37 million names
    http://earth-info.nga.mil/gns/html/

- The World Gazetteer provides a comprehensive set of population data and related statistics:
  http://www.world-gazetteer.com/

- Wikipedia

- Linked data: http://linkeddata.org/
Write rules to recognize names

R1: if features then person
R2: if features then location
R3: if features then organization

Features like capitalization: (not enough)

- Full-string=U.S. → Location
- Full-string=I.B.M → organization
Lexical & Context Features

- **Set of spelling features**
  - Full-string=x (full-string=Maro Cooper)
  - Contains(x) (contains(Maco))
  - Allcap1 (IBM)
  - Allcap2 (N.Y.)
  - Nonalpha=x A.T.&T. nonalpha=.&

- **Set of context features**
  - Context=x (context=president)
  - Context-type=x (prep or apposition)
Parsing-based Features

- **Has_Predicate**: from logical subject to verb
  
  e.g. *He said she would want him to join* →
  
  has: Has_Predicate(*say*), she: Has_Predicate(*want*), him: Has_Predicate(*join*)

- **Has_Amod**: from noun to its adjective modifier
  
  e.g. *He is a smart, handsome young man* → man: Has_AMod(*smart*)

- **Possess**: from the possessive noun-modifier to head noun
  
  e.g. *His son was elected as mayor of the city* → his: Possess(*son*), city: Possess(*mayor*)
Example: some rules for person

Possess(wife) → PER
Possess(brother) → PER
Possess(daughter) → PER
Possess(bravery) → PER
Possess(father) → PER

Has_Predicate(divorce) → PER
Has_Predicate(remarry) → PER
Some rules for Location

Possess(concert_hall) → LOC
Possess(mayor) → LOC

Has_AMod(coastal) → LOC
Has_AMod(northern) → LOC
Has_AMod(eastern) → LOC
Has_AMod(northeastern) → LOC
Typical **lexical features** in a named entity recognition task (for example) (candidate entity name $i$ that occur in the context window of $l$ words)

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>VALUE TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short type</td>
<td>Boolean</td>
<td>True if $i$ matches the short type $j$; False otherwise.</td>
</tr>
<tr>
<td>POS</td>
<td>Nominal</td>
<td>Part-of-speech tag of the syntactic head of $i$.</td>
</tr>
<tr>
<td>Context word</td>
<td>Boolean or real value between 0 and 1; Or nominal.</td>
<td>True if the context word $j$ occurs in the context of $i$; False otherwise; If a real value is used, it indicates the weight of the context word $j$. Alternatively, the context word feature can be represented as one feature with nominal values.</td>
</tr>
<tr>
<td>POS left</td>
<td>Nominal</td>
<td>POS tag of a word that occurs to the left of $i$.</td>
</tr>
<tr>
<td>POS right</td>
<td>Nominal</td>
<td>POS tag of a word that occurs to the right of $i$.</td>
</tr>
<tr>
<td>Morphological prefixes/suffixes</td>
<td>Nominal</td>
<td>Prefix or suffix of $i$.</td>
</tr>
</tbody>
</table>
For example: person name

*Microsoft spokesman *John Smith* is a popular man.*

<table>
<thead>
<tr>
<th>Features</th>
<th>Values</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full string</td>
<td>True</td>
<td>The first letter of each word is capitalized.</td>
</tr>
<tr>
<td>Allcap 1</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>POS</td>
<td>np</td>
<td>Brown corpus: nn for singular common nouns,nns for plural common nouns, np for singular proper nouns.</td>
</tr>
<tr>
<td>Contain “Mr”</td>
<td>“Dr”</td>
<td>No</td>
</tr>
<tr>
<td>Contain [CEO, CFO,spokesman,...]</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>POS left</td>
<td>nn</td>
<td></td>
</tr>
<tr>
<td>POS right</td>
<td>vb</td>
<td></td>
</tr>
<tr>
<td>Morphological prefix or suffix</td>
<td></td>
<td>Prefix: co (joint, with), pro(for, forward), re(again,back)</td>
</tr>
</tbody>
</table>
For example: person name (cont.)

- Microsoft spokesman John Smith is a popular man.
  
  **Lexical**: cap lower cap cap lower lower lower lower punt
  
  **POS**: nnp nn nnp nnp vb dt adj nn punt

- Feature selection: use which features to identify person name.

- Different methods have different features.
## Typical features in a single-document noun phrase coreference resolution task

- Two candidates $i$ and $j$, where $i < j$ in terms of word position

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>VALUE TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number agreement</td>
<td>Boolean</td>
<td>True if $i$ and $j$ agree in number; False otherwise.</td>
</tr>
<tr>
<td>Gender agreement</td>
<td>Boolean</td>
<td>True if $i$ and $j$ agree in gender; False otherwise.</td>
</tr>
<tr>
<td>Alias</td>
<td>Boolean</td>
<td>True if $i$ is an alias of $j$ or vice versa; False otherwise.</td>
</tr>
<tr>
<td>Weak alias</td>
<td>Boolean</td>
<td>True if $i$ is a substring of $j$ or vice versa; False otherwise.</td>
</tr>
<tr>
<td>POS match</td>
<td>Boolean</td>
<td>True if the POS tag of $i$ and $j$ match; False otherwise.</td>
</tr>
<tr>
<td>Pronoun $i$</td>
<td>Boolean</td>
<td>True if $i$ is a pronoun; False otherwise.</td>
</tr>
<tr>
<td>Pronoun $j$</td>
<td>Boolean</td>
<td>True if $j$ is a pronoun; False otherwise.</td>
</tr>
<tr>
<td>Appositive</td>
<td>Boolean</td>
<td>True if $j$ is the appositive of $i$; False otherwise.</td>
</tr>
<tr>
<td>Definiteness</td>
<td>Boolean</td>
<td>True if $j$ is preceded by the article “the” or a demonstrative pronoun; False otherwise.</td>
</tr>
<tr>
<td>Grammatical role</td>
<td>Boolean</td>
<td>True if the grammatical role of $i$ and $j$ match; False otherwise.</td>
</tr>
<tr>
<td>Proper names</td>
<td>Boolean</td>
<td>True if $i$ and $j$ are both proper names; False otherwise.</td>
</tr>
<tr>
<td>Named entity class</td>
<td>Boolean</td>
<td>True if $i$ and $j$ have the same semantic class (e.g., person, company, location); False otherwise.</td>
</tr>
<tr>
<td>Discourse distance</td>
<td>Integer $\geq 0$</td>
<td>Number of sentences or words that $i$ and $j$ are apart.</td>
</tr>
</tbody>
</table>
Typical features in a cross-document noun phrase coreference resolution task

- i and j occur in different documents.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context word</td>
<td>Boolean or real value be-</td>
<td>True if the context word $k$ occurs in the context of $i$ and $j$; False otherwise; If a real value is used, it indicates the weight of the context word; Proper names, time and location expressions in the context might receive a high weight.</td>
</tr>
<tr>
<td>Named entity class</td>
<td>Boolean</td>
<td>True if $i$ and $j$ have the same semantic class (e.g., person, company, location); False otherwise.</td>
</tr>
<tr>
<td>Semantic role</td>
<td>Boolean</td>
<td>True if the semantic role of $i$ matches the semantic role of $j$; False otherwise.</td>
</tr>
</tbody>
</table>
Pattern Models

Predicate-Argument Model (SVO)

Chains: a path between a verb node and any other node in a dependency tree passing through zero or more intermediate nodes.

Linked Chains: a pair of chains which share the same verb but no direct descendants.

Sub-tree: any subtree of a dependency tree can be used as an extraction pattern.
Pattern Examples

SVO
[V/hire] (subj[N/Acme Inc.] + obj[N/Mr Smith])
[V/replace] (obj[N/Mr Bloggs])

Chains
[V/hire] (subj[N/Acme Inc.])
[V/hire] (obj[N/Mr Smith])
[V/hire] (obj[N/Mr Smith] (as [N/CEO]))
[V/hire] (obj[N/Mr Smith] (as [N/CEO] (gen[N/their])))
[V/hire] (obj[N/Mr Smith] (as [N/CEO] (mod[A/new])))
[V/hire] (vpsc_mod[V/replace])
[V/hire] (vpsc_mod[V/replace] (obj[N/Mr Bloggs]))
[V/replace] (obj[N/Mr Bloggs])

Linked Chains
[V/hire] (subj[N/Acme Inc.] + obj[N/Mr Smith])
[V/hire] (subj[N/Acme Inc.] + obj[N/Mr Smith] (as [N/CEO]))
[V/hire] (obj[N/Mr Smith] + vpsc_mod[V/replace] (obj[N/Mr Bloggs]))
Summarization

- What are the named entities?
- How to identify named entities?
  - Find features: indicative, informative
  - Build patterns: using many features. Not too general, not too specific.
  - Apply patterns

- Rule-based methods: build patterns by human beings.
Other References

1) Language independent NE Recognition :
   http://www.cnts.ua.ac.be/conll2003/ner/

2) global gazetteer version 2.1
   http://www.fallingrain.com/world/

3) Rohini K.Srihari et al,”InfoXtract; A customizable Intermediate Level Information Extraction Engine

Classroom Discussion

☐ How to identify organization name?

☐ 清远绿由环保科技有限公司主要从事固体废物无害化处置和资源化利用项目。在展台上，记者看到了工业污泥、陶瓷废渣处理加工制成的环保科技砖块，将废胶“变废为宝”制成的各类毛刷等。国家工信部是中小企业行政主管部门，来自工信部的总工程师朱宏任介绍说，国家出台的这份文件既考虑解决了小型微型企业当前面临的生产经营困难，又注重引导企业增强内生动力，还提出了支持企业长期平稳健康发展的长效机制。说到中小企业创新的问题，兴业银行首席经济学家鲁政委认为，中小企业融资最难的时候已经过去了，恰恰相反，中小企业投融资难都得到了比较好的改善，所有中小企业最缺的是优质客户，在技术、手段、平台上还需要更具体的、务实的创新。

2014/10/11
机构名称的组成：公司特征词（后缀判断）+名称组成词（前部判断）

公司特征词是有限的，可以放在字典中，如国家机关名（部委），教育科研机构（大学），公共设施及场所（公园，体育馆），医疗机构，商业机构，社会组织，体育组织，体育组织，娱乐场所等。

名称组成词包括：地名，人名（李宁体育公司），学科（电子科技），研究生产经营对象（五金工具批发市场/商店，软件研究所/公司），音译词（协和医院），创办，工作方式（集团，股份）。

2014/10/11
Rule Deduction

- Org = [ModifierWord] + [FeatureWord]
- FeatureWord = 公司 | 大学 | 机构 | ...
- POS (ModifierWord) = adj | np | nnp | nz | vn | ...

1. n+nz+ Company (Corp. Ltd.)
2. ns+nz+University (college, school)
3. adj+Foundation (Agency)
Some Solutions:

- **Lexicon**: keep famous company names
- **Parsing-based Rules**:
  - Has_AMod(*advisory*) → ORG
  - Has_AMod(*non-profit*) → ORG
  - Possess(*ceo*) → ORG
  - Possess(*operate loss*) → ORG
  - Has_AMod(*multinational*) → ORG
  - Has_AMod(*non-governmental*) → ORG
- **Heuristic Clues**:
  - It is consecutive, not cross sentence boundary or any punctuation.
  - It appears often.
Some features of organization name...

- Context word: {董事长 | 经理 | 发言人 }
- POS Left: {}
- POS right: {}

For example:
- 河南省/ns商丘市/ns中级/b人民法院/nz作出/v一审/n判决/vn