Introduction to Human Language Technologies

1. Document structure
Outline

1. Document structure
   - Searching textual zones
   - Tokenization
   - Sentence splitting

2. Language identification
Outline

1 Document structure
   - Searching textual zones
   - Tokenization
   - Sentence splitting

2 Language identification
Document types

- Documents containing text:
  - Structured documents (e.g., web pages being tables)
  - Semi-structured documents (e.g., web pages containing pieces of plain text, figures and tables)
  - Documents with plain text only (e.g., text files, emails, tweets, oral transcripts)

Accessing to plain text contained in web pages may be relevant.
XML Parsers

- Transform an XML/HTML/XHTML document into a tree of standard objects.
- Provide an interface to manage that tree.
- Textual zones in the document can be extracted from that tree using the interface.

```xml
<?xml version="1.0"?>
<doc type="novel" title="The green apple">
  <chapter id="1">
    <p>There are lots of trees in Amsteel Hill. I remember going there and spend all the morning climbing those trees, trying to get as many apples as possible.</p>
    <p>James always wanted to come with me but he was too young to get climbing.</p>
    ...
  </chapter>
</doc>
```

Using `ElementTree.py`

```python
for c in root:
    lp=c.findall('p')
    for p in lp:
        print p.text
```
Outline

1 Document structure
   - Searching textual zones
   - Tokenization
   - Sentence splitting

2 Language identification
Goal of tokenization

- Goal: split plain text into *basic units*
- Use: IR tasks, text categorization, sentence splitting, language identification, text normalization . . .
- Different *basic units* depending on the task,
  - *Naïve* tokenizations: split by blanks and punctuation marks occurring after alphanum-string.
  - Complex tokenizations: names, clitics, abbreviations, *collocations* . . .
Goal of tokenization

- Goal: split plain text into *basic units*
- Use: IR tasks, text categorization, sentence splitting, language identification, text normalization . . .
- Different *basic units* depending on the task,
  - *Naïve* tokenizations: split by blanks and punctuation marks occurring after alphanum-string.
  - Complex tokenizations: names, clitics, abbreviations, *collocations* . . .

Relevant definitions:

*Word N-gram*: sequence of words occurring in a text

*Collocation*: sequence of words that frequently occur together. Ex: "break a leg", "On the one hand"
## Examples of tokenization

<table>
<thead>
<tr>
<th>Blanks</th>
<th>outer punct.</th>
<th>Abbr.</th>
<th>Clitics</th>
<th>Colloc.</th>
<th>text normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Of course</td>
<td>Of course</td>
<td>Of course</td>
<td>Of course</td>
<td>Of course</td>
<td>Of course</td>
</tr>
<tr>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
</tr>
<tr>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
</tr>
<tr>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;Daily,</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
</tr>
<tr>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mister</td>
</tr>
<tr>
<td>John Smith...”</td>
<td>John Smith</td>
<td>John Smith</td>
<td>John Smith</td>
<td>John Smith</td>
<td>John_Smith</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
## Examples of tokenization

<table>
<thead>
<tr>
<th>Blanks</th>
<th>outer punct.</th>
<th>Abbr.</th>
<th>Clitics</th>
<th>Colloc.</th>
<th>text normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Of</td>
<td>Of</td>
<td>Of</td>
<td>Of</td>
<td>Of_course</td>
<td>Of_course</td>
</tr>
<tr>
<td>course</td>
<td>course</td>
<td>course</td>
<td>course</td>
<td>course</td>
<td>course</td>
</tr>
<tr>
<td>I'll</td>
<td>I'll</td>
<td>I' ll</td>
<td>I'I'll</td>
<td>I'I'll</td>
<td>I'I'll will</td>
</tr>
<tr>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
</tr>
<tr>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>,</td>
<td>,</td>
<td>,</td>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>”Daily,</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
</tr>
<tr>
<td></td>
<td>,</td>
<td>,</td>
<td>,</td>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>Mr.</td>
<td>Mr</td>
<td>Mr</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>John</td>
<td>John</td>
<td>John</td>
<td>John</td>
<td>John</td>
<td>John_Smith</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
<td>Smith</td>
<td>Smith</td>
<td>Smith</td>
<td>Smith</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>”</td>
<td>”</td>
<td>”</td>
<td>”</td>
<td>”</td>
<td>”</td>
</tr>
</tbody>
</table>
### Examples of tokenization

<table>
<thead>
<tr>
<th>Blanks</th>
<th>outer punct.</th>
<th>Abbr.</th>
<th>Clitics</th>
<th>Colloc.</th>
<th>text normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Of course</td>
<td>Of course</td>
<td>Of course</td>
<td>Of course</td>
<td>Of_course</td>
<td>Of_course</td>
</tr>
<tr>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll</td>
<td>I’ll will</td>
</tr>
<tr>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
<td>go</td>
</tr>
<tr>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
<td>to</td>
</tr>
<tr>
<td>U.P.C.</td>
<td>U.P.C</td>
<td>U.P.C</td>
<td>U.P.C</td>
<td>U.P.C</td>
<td>Universitat. ...</td>
</tr>
<tr>
<td>’’</td>
<td>’’</td>
<td>’’</td>
<td>’’</td>
<td>’’</td>
<td>’’</td>
</tr>
<tr>
<td>”Daily,</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
<td>Daily</td>
</tr>
<tr>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mr.</td>
<td>Mister</td>
</tr>
<tr>
<td>John Smith...”</td>
<td>John Smith</td>
<td>John Smith</td>
<td>John Smith</td>
<td>John Smith</td>
<td>John_Smith</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Outline

1. Document structure
   - Searching textual zones
   - Tokenization
   - Sentence splitting

2. Language identification
Goal of sentence splitting

- **Goal:** Recognition of sentence boundaries in plain text (e.g., '. ' '?' '! '...').
- **Language-dependent task**
  - Ex: German: "Mein 2. Semester kommt bald zu Ende."
  - Ex: Traditional chinese?
- **Domain-dependent task**
  - Ex: "It is expressed as \((x=1)? T.add(-') : T.add(x)\)."
- **Methods:**
  - Hand-crafted rules
  - Machine learning methods
- **Input:**
  - *Naïve* tokenization that depends on the particular method.
  - For simplicity, we will assume *blanks+outer_punctuation*
    
    " I’ll go to U.P.C. " Daily, Mr. John Smith...
    
    → " I’ll go to U.P.C. " Daily, Mr. John Smith..."
Main problems:

- Abbreviations and acronyms (most difficult one)
  
  Ex: "I will meet with Mr. Smith to talk about it."
  
  Ex: "Lisa run 25 km. She ended up in N.Y."

How to detect them?

- Ellipsis
  
  Ex: "There’re different methods (A, B, . . . ) but . . . ”

- Internal quotation
  
  Ex: ” ’Stop!’ he shouted.”

- Ordinal numbers (German)

- Special cases:
  
  Ex: ” We have some variables. x stands for the weight,”
Hand-crafted rules for sentence splitting

- Specific hand-crafted rules for specific cases
  - Abbreviation classes (Lists of abbreviations) (month name, unit-of-measure, title, address name, . . .)
    Ex: TITLE=('Mr', 'Mrs', 'Dr', . . .)
  - Regular expressions for general cases, abbreviations, ellipsis, . . .
    Ex: / ([?!]+) / → t ∈ s_boundary
    Ex: / (\./\./\.) [A-Z]/ → t ∈ s_boundary
    Ex: / ([?!.]) [A-Z]/ → t ∈ s_boundary
    Ex: / ($TITLE) \. / → t ∉ s_boundary
    Ex: / [A-Z] \. / → t ∉ s_boundary

- Problem:
  - Highly expensive adaptation to new languages (rules and abbreviation classes)
The most frequently used (ME, SVM, CRF, . . .)

Require manually annotated corpora. Commonly, $e^+, e^- = ['.','!','?']$ and some preceding and following tokens

Represent each $e$ as a set of features. Depends on the approach, the language and the domain, although normally they tend to be binary features.

Problem:

Require very large sets of examples (tens of thousands to hundreds of thousands)
Supervised ML for sentence splitting

- Examples of features used in the state of the art
  tok-1_\text{x}: 1st token before "." is \text{X}
  tok-2_\text{x}: 2nd token before "." is \text{X}
  tok+1_\text{x}: 1st token after "." is \text{X}
  len_tok-1_\text{x}: length of 1st token before "." is \text{X}
  len_tok-2_\text{x}: length of 2nd token before "." is \text{X}
  len_tok+1_\text{x}: length of 1st token after "." is \text{X}
  [up|lo|cap|num]_tok-1: 1st token before "." is Upper, Lower, CAP, Numbers
  [up|lo|cap|num]_tok-2: same for 2nd token before "."
  [up|lo|cap|num]_tok+1: same for 1st token after "."
  class_tok-1_\text{x}: abbreviation class of 1st token before "." is \text{X}
  ...

Supervised ML for sentence splitting

Example of annotation and binary features extraction

I’ll go to U.P.C “ Daily, Mr John Smith ... ”

e+ tok-1_U.P.C len_tok-1_3 CAP_tok-1 tok-2_to len_tok-2_2 lo_tok-2 tok+1_” len_tok+1_1
e- tok-1_Mr len_tok-1_2 up_tok-1 tok-2, len_tok-2_1 class_tok-1_title tok+1_John len_tok+1_4 up_tok+1
Unsupervised ML for sentence splitting

- Based on corpus statistics
- Easily adaptable to new languages
  - They require large unannotated training corpora
- Mainly focus on abbreviations and ellipsis
- Heuristics and statistics calculated from the training corpus to decide:
  1. Which tokens are abbreviations?
  2. When the final period of the elements is a sentence boundary?
- Example: Punkt [Kiss and Strunk, 2006]
Unsupervised ML for sentence splitting

1. **Punkt: Is token $t$ considered an abbreviations?**
   Measured by considering the following heuristics:
   - $t' = <t, .>$ should be a collocation
   - the length of $t$ should be short
   - $t$ could include periods (acronyms)
   - $t$ is not ordinary word preceding a period most of the times. (e.g., verbs in Turkish)
Unsupervised ML for sentence splitting

1. **Punkt: Is token $t$ considered an abbreviations?**

   Measured by considering the following heuristics:
   - $t' = <t, .>$ should be a collocation
   - the length of $t$ should be short
   - $t$ could include periods (acronyms)
   - $t$ is not ordinary word preceeding a period most of the times. (e.g., verbs in Turkish)

2. **Punkt: Is the final period of abbreviation $t' = <t, .>$ considered sentence boundary?**

   Either one of the following heuristics must be true:
   - $t'' = following(t')$ is a frequent sentence (from [1]) starter
   - $t''$ is uppercase, occurs at least once in lowercase in the training corpus but never in uppercase inside sentences (from [1])
Exercise

Explain why Punkt fails (red) or not (blue) with the following texts:

- ” ”Good night!”, said Laura. ”
- ” Abbrev. is a common abbreviation of abbreviation. ”
- ” We are meeting with our mr. You are late! ”
- ” We are meeting with our Mr. However, we’ll finish soon.”

Demo sentence splitters:
http://text-processing.com/demo/tokenize/
Outline

1. Document structure
   - Searching textual zones
   - Tokenization
   - Sentence splitting

2. Language identification
Goal of language identification

- Can be seen as a particular classification problem.
- Given a document, $d$, and a set of languages, $L = \{l_1, \ldots, l_k\}$, assign $l_i$ to $d$.
- Method:
  - $\hat{d} = \text{representation}(d)$
  - $M(\hat{d}) \rightarrow l_i$
- Model $M$ can be learned from training corpus $T = \{T_i\}_{1 \ldots k}$ where $T_i = \{d_x | d_x \text{ written in } l_i\}$:
  - Supervised Machine Learning methods
  - Statistical Language models

Language models for language identification

Method with language models:

\[ M = \{ P^i_l \}_{i \in L} \]
\[ P^i_l(\hat{d}) : \text{probability of } \hat{d} \text{ to belong to } l_i \]

\[ l_i = \arg \max_{l \in L} (P^l(\hat{d})) \]

\[ P^l_i(\hat{d}) \approx P^{T_i}(\hat{d}) : \text{probability of } \hat{d} \text{ observing data from } T_i \]
Language models for language identification

Method with language models:

\[ M = \{P^l_i\}_{l_i \in L} \]

\[ P^l_i(\hat{d}) \]: probability of \( \hat{d} \) to belong to \( l_i \)

\[ l_i = \text{argmax}_{l \in L}(P^l(\hat{d})) \]

\[ P^l_i(\hat{d}) \approx P^{T_i}(\hat{d}) \]: probability of \( \hat{d} \) observing data from \( T_i \)

1. Which is the representation \( \hat{d} \)?
2. How is \( P^{T_i}(\hat{d}) \) computed?
Language models for language identification

Method with language models:

\[ M = \{P^l_i\}_{l_i \in L} \]

\[ P^l_i(\hat{d})\]: probability of \( \hat{d} \) to belong to \( l_i \)

\[ l_i = \arg\max_{l \in L} (P^l(\hat{d})) \]

\[ P^l_i(\hat{d}) \approx P^{T_i}(\hat{d})\]: probability of \( \hat{d} \) observing data from \( T_i \)

1. Which is the representation \( \hat{d} \)?
2. How is \( P^{T_i}(\hat{d}) \) computed?

They depend on the particular type of model.

Most frequently used: unigram language models
Unigram language models for language identification

1. **Which is the representation \( \hat{d} \)?**

   \( \hat{d} = e_1, \ldots, e_s \) being the occurrences of unigrams:
   - Words (after *Naïve* tokenization) or
   - Characters \( n \)-grams (tokenization is not required)
     - \( n \) fixed (the most frequently used) or
     - \( n \) variable (improves accuracy, lower efficiency)
Unigram language models for language identification

1. Which is the representation $\hat{d}$?
   $\hat{d} = e_1, \ldots, e_s$ being the occurrences of unigrams:
   - Words (after Naïve tokenization) or
   - Characters $n$-grams (tokenization is not required)
     - $n$ fixed (the most frequently used) or
     - $n$ variable (improves accuracy, lower efficiency)

2. How is $P^T_i(\hat{d})$ computed?
   Each $e_j$ is independent from the rest
   \[
P^T(\hat{d}) = P^T(e_1, \ldots, e_s) = \prod_{j=1}^{s} P^T(e_j)
   \]
   \[
   \log P^T(\hat{d}) = \sum_{j=1}^{s} \log P^T(e_j)
   \]
   Possible estimators of $P^T(e_j)$:
   - Maximum Likelihood Estimator (MLE)
   - Smoothing techniques.
Unigram language models for language identification

Maximum Likelihood Estimator

\[ P^T(e_j) \approx P^T_{\text{MLE}}(e_j) = \frac{c_T(e_j)}{N_T} \]

- \( c_T(x) \): \#observed occurrences of \( x \) in training corpus \( T \)
- \( N_T \): \#observed occurrences of elements in training corpus \( T \)
Unigram language models for language identification

Maximum Likelihood Estimator

\[
P^T(e_j) \approx P^T_{MLE}(e_j) = \frac{c_T(e_j)}{N_T}
\]

- \(c_T(x)\): #observed occurrences of \(x\) in training corpus \(T\)
- \(N_T\): #observed occurrences of elements in training corpus \(T\)

Problem: data sparseness. Unseen \(e_j\) causes the model to fail. MLE is unsuitable for NLP.
Unigram language models for language identification

Maximum Likelihood Estimator

\[ P^T(e_j) \approx P_{MLE}^T(e_j) = \frac{c_T(e_j)}{N_T} \]

- \( c_T(x) \): #observed occurrences of \( x \) in training corpus \( T \)
- \( N_T \): #observed occurrences of elements in training corpus \( T \)

Example:

\[ P^{[en]}('The doctor tell us about his quadriplegia')? \]

\[ c^{[en]}('quadriplegia') = 0 \implies P_{MLE}^{[en]}('quadriplegia') = 0 \]
\[ \implies P^{[en]}('The doctor tell us about his quadriplegia') = 0 \]
Smoothing Techniques:

Keep some probability mass for $e_j$ unseen in $T_i$
E.g., Lidstone’s Law (LID)

$$P^T(e_j) \approx P^T_{LID}(e_j) = \frac{c_T(e_j) + \lambda}{N_T + \lambda B} \quad \text{usually, } \lambda = 0, 5$$

$B$: #bins (potentially observable unigrams)
Exercise

Suppose we have a Language Identifier for English and Catalan, based on unigram language models with words and the following statistics:

<table>
<thead>
<tr>
<th>$w_i$</th>
<th>a</th>
<th>he</th>
<th>mail</th>
<th>sent</th>
<th>to</th>
<th>mordorian</th>
</tr>
</thead>
<tbody>
<tr>
<td>English language model [en]</td>
<td>$c_{[en]}(w_i)$</td>
<td>17.000</td>
<td>10.000</td>
<td>3.900</td>
<td>850</td>
<td>25.000</td>
</tr>
<tr>
<td>$N_{[en]}=1.300.000$</td>
<td>$B_{[en]}=22.600$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catalan Language model [ca]</td>
<td>$c_{[ca]}(w_i)$</td>
<td>21.000</td>
<td>11.900</td>
<td>420</td>
<td>910</td>
<td>750</td>
</tr>
<tr>
<td>$N_{[ca]}=1.100.000$</td>
<td>$B_{[ca]}=36.800$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Compute $P_{[en]}$ and $P_{[ca]}$ using MLE and LID for the following texts:
  - ”he”
  - ”he sent a”
  - ”he sent a mail”
  - ”he sent a mail to a mordorian”
- What language is identified by each estimator for each of the previous texts?
- Explain the effects of the text size