Introduction to Natural Language Processing (NLP)

Most of the slides are rielaborated from Jannik's course on IR

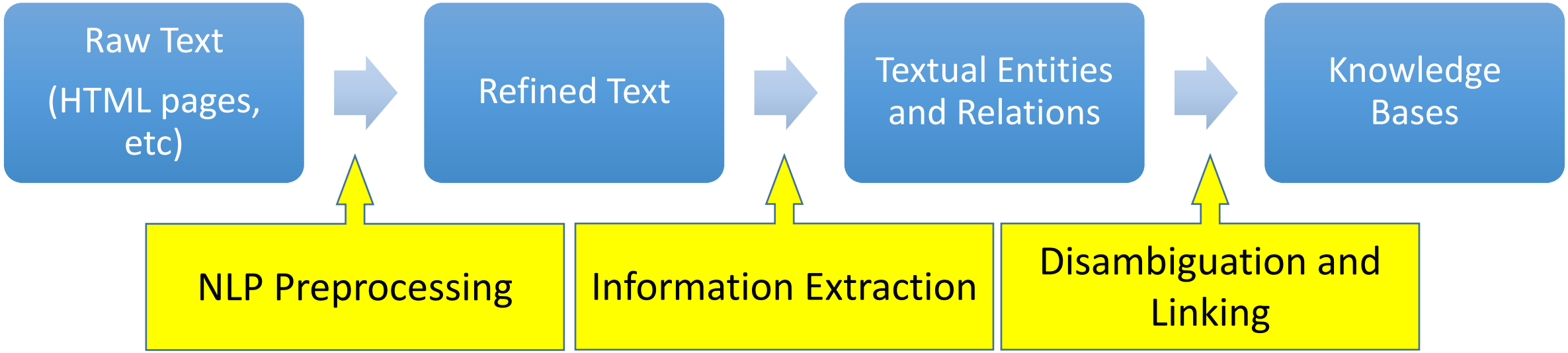
(http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/teaching/winter-semester-201617/information-extraction/)

Also (http://web.stanford.edu/~jurafsky/slp3/)

VU VRIJE UNIVERSITEIT AMSTERDAM

What is knowledge acquisition?

Knowledge acquisition: process to extract knowledge (to be integrated into knowledge bases) from unstructured text or other data



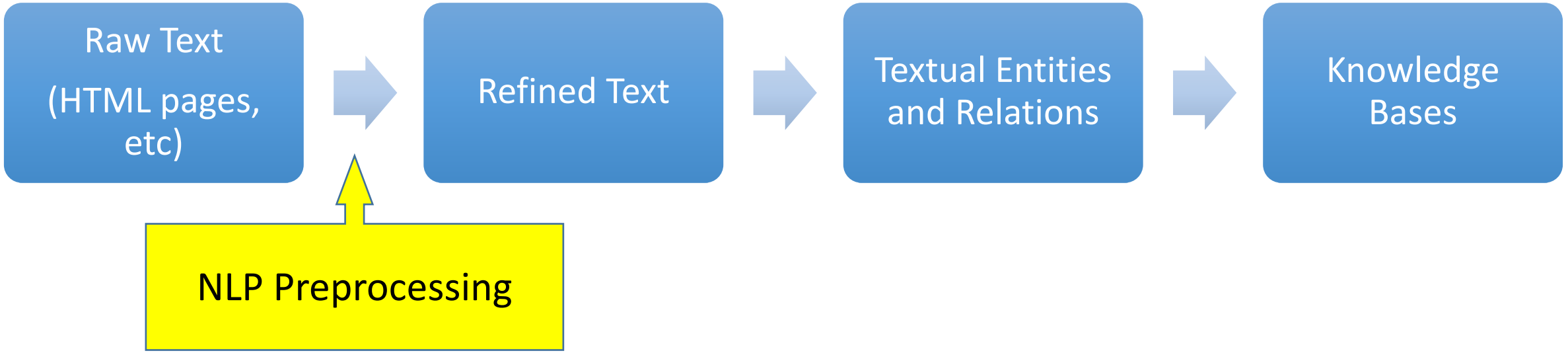
Types of extraction

- In this course we will discuss techniques to extract entities and relations between them from unstructured data

- Another important form of extraction consists of detecting *events* and other *temporal expressions*. We are not going to talk about them

What is knowledge acquisition?

Knowledge acquisition: process to extract knowledge (to be integrated into knowledge bases) from unstructured text or other data



VU VRIJE UNIVERSITEIT AMSTERDAM

Natural Language Processing (NLP)

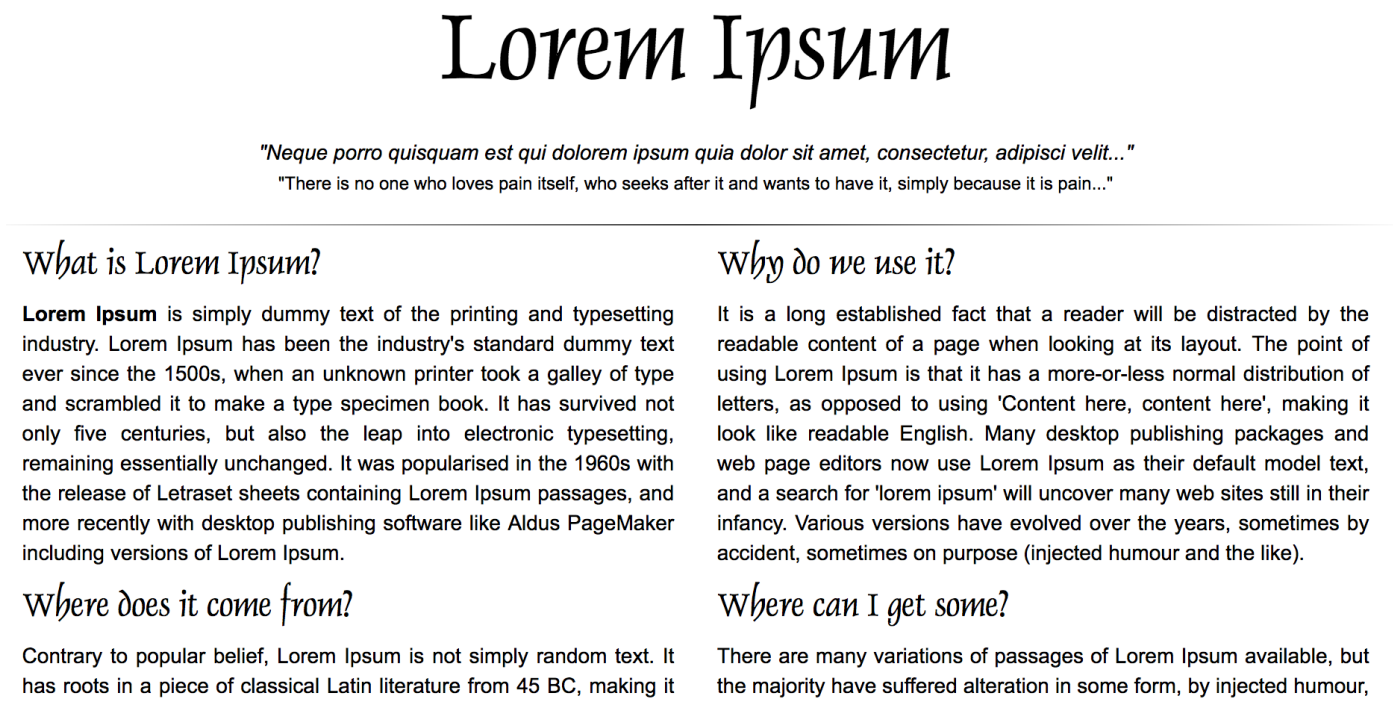
VU VRIJE UNIVERSITEIT AMSTERDAM

Why is NLP important?

- **Unstructured data**
  - Typically refers to “text”
- **Typical distinction**
  - Structured data => “databases”
  - Unstructured data => “information retrieval”
- Actually: **Semi-structured data**
  - Almost always some structure: title, bullets

NLP preprocessing

- Before we can use some text, we must pre-process it
- Standard tasks
  - **Tokenization**
  - **Stemming** or **lemmatization**
  - **Stopword removal**
  - **POS tagging**
  - **Parsing**



Lorem Ipsum

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."

"There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain..."

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it?

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Where does it come from?

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it

Where can I get some?

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour,

VU VRIJE UNIVERSITEIT AMSTERDAM

Tokenization

The task

- Given a character sequence, split it into pieces called tokens
- Tokens are often loosely referred as terms/words

Type/Token

- **Token:** instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit
- **Type:** class of all tokens containing the same character sequence

Tokenization

- **Example:** A rose is a rose is a rose
- How many tokens? 8
- How many types? 3 ({a, is, rose})
- **Set-theoretical view:**
- Tokens => multiset
- Types => set

Tokenization

- **Example:** Mr. O'Neill thinks rumors about Chile's capital aren't amusing
- **Simple strategy**
  - Split at white spaces and hyphens
  - Split on all non-alphanumeric characters
  - Mr | o | neill | thinks | rumors | about | chile | s | capital | aren| t | amusing
- **Is that good?**
  - Many alternatives! O | neill – oneill – neill – o'neill – o' | neill
- Even such simple task is not trivial!

Tokenization

- **Queries and documents have to be preprocessed identically**
  - Tokenization determines which queries match
  - Guarantees that sequence of characters in query matches the same sequence in text
- **Further issues**
  - What about hyphens? Co-education, drag-and-drop
  - What about names? Los Angeles, San Francisco
  - Tokenization is language-specific
    - Asian languages 这是几个单词序列
    - Noun compounds are not separated in German: “Lebensversicherungsgesellschaftangesteller” vs. “life insurance company employee”

Tokenization

- State-of-the-art
  - Stanford Tokenizer (http://nlp.stanford.edu/software/tokenizer.shtml)
  - Apache OpenNLP (https://opennlp.apache.org/)
  - NLTK (http://www.nltk.org/)

Lemmatization and stemming

After tokenization, following tasks are

- Lemmatization
- Stemming
- Stopword removal

Lemmatization and stemming

- Two tasks, same goal

What's the difference?

Lemma and lemmatization

- Goal: reduce inflectional forms (all variants of a “word”) to base form

Examples

- Am, are, be, is => be
- Car, cars, car’s, cars’ => car

Lemmatization

- Proper reduction to dictionary headword form

Lemma

- Dictionary form of a set of words

Stem and Stemming

Idea

- Reduce terms to their “roots”

Examples

- Are => ar
- Automate, automates, automatic, automation => automat

Stemming

- Suggests crude affix chopping

Stem

- Root form of a set of words (not necessarily a word itself)

Stemming and Lemmatization

The boy’s cars are different colors

- **Lemmatized:** the | boy | car | be | different | color
- **Stemmed:** the | boy | car | ar | differ | color

Stemming and Lemmatization

For example compressed and compression are both accepted as equivalent to compress

- **Lemmatized:** for | example | compress | and | compression | be | both | accept | as | equivalent | to | compress
- **Stemmed:** for | exampl | compress | and | compress | are | both | accept | as | equival | to | compress

Stemming

Popular stemmers

- Porter's algorithm (http://tartarus.org/martin/PorterStemmer/)
- Snowball (http://snowballstem.org/demo.html)

What's better for IR? Stemming or lemmatization?

- No golden rule. You just have to try it out

Stop Words

Stop words

- Have little semantic content
- Are extremely frequent
- Occur in almost each document, i.e., are not discriminative

Example of a stop word list

- a, an, and, are, as, at, be, by, for, from, has, he, in
- is, it, its, of, on, that, the, to, was, were, will, with

What types of words are these?

Stop Word Removal

Idea

- Based on stop list, remove all stop words, i.e., stop words are not part of IR system's dictionary
- Saves a lot of memory
- Makes query processing much faster

Trend (in particular in web search)

- No stop word removal
  - there are good compression techniques
  - There are good query optimization techniques

Stop Word Removal

In some cases stop words are needed

- King of Norway
- Let it be
- To be or not to be

Part-of-Speech

- **Problem:** Assign to each token a label that indicates what it is
- Function words: used to make sentences grammatically correct
- Content words: carry the meaning of a sentence

Function words

- Auxiliary verbs
- Prepositions
- Conjunctions
- Determiners
- Pronouns

Content words

- Nouns
- Verbs
- Adjectives
- Adverbs

See Ch. 9 of Speech and Language Processing, Dan Jurafsky and James H. Martin. http://web.stanford.edu/~jurafsky/slp3/9.pdf

Part-of-Speech

How many parts-of-speech are there?

- Between 8 (English) and hundreds (depending on application/other languages) of different parts-of-speech
- What's useful depends on the application and language

One word, one part-of-speech?

- Example: Can we can fish in a can?
- Can: auxiliary, verb, noun

Part-of-Speech Tagging

Part-of-speech tags

- Allow for higher degree of abstraction to estimate likelihoods

What's the likelihood of:

- "an amazing" – is followed by "goalkeeper"
- "an amazing" – if followed by "scored"
- "determiner adjective" – is followed by "noun"
- "determiner adjective" – is followed by "verb"

Part-of-Speech Tagging

Automatic assignment of part-of-speech tags

- Penn Treebank tagset is one of the most used list of tags
- 36 tags (List)

Way to go

- Input: sequence of (tokenized) words
- Output: chain of tokens with their part-of-speech tags
- Goal: most likely part-of-speech tags for the sequence
- A typical classification problem

Part-of-Speech Tagging

Is it hard?

- Most words in English are not ambiguous
- Most occurring words in English are ambiguous

Today's taggers

- About 97% accuracy (but there is a difference between domain and domain), but baseline (=most frequent choices) is already 90%!

Part-of-Speech Tagging

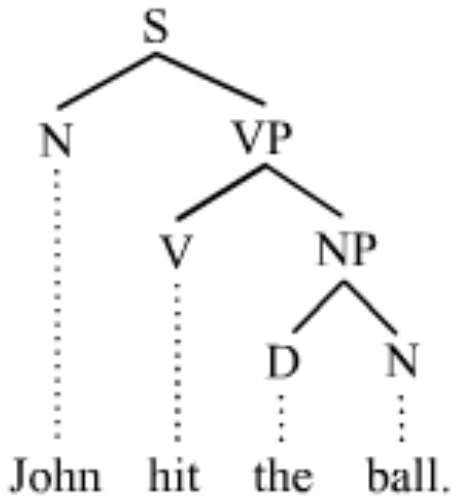
How do they work?

- rule-based taggers (e.g. Brill's tagger [1], one of the first and most widely used ones)
- Stochastic taggers. These are the most used and rely on HMM (Hidden Markov Models)
- Comparison of several methods available at [2]

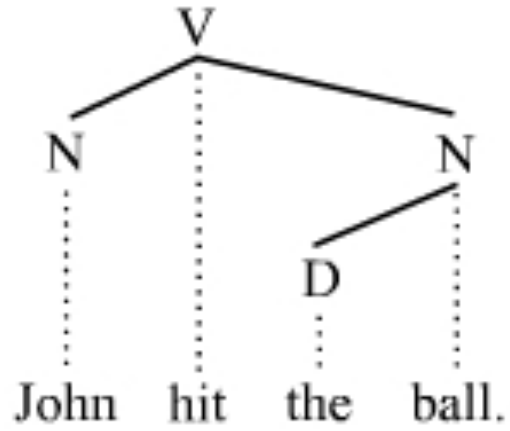
[1] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the workshop on Speech and Natural Language*, 1992, pp. 112–116.
[2] https://www.aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

Other NLP tasks

- There are other tasks in typical NLP pipelines
  - **Parsing:** Construct a tree that represents the syntactic structure of the string according to some grammars



Constituency-based parse tree



Dependency-based parse tree

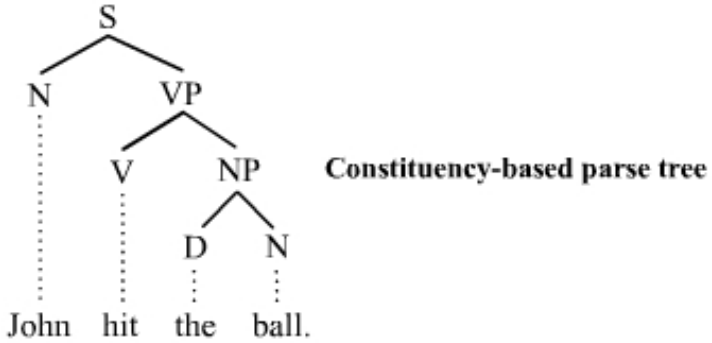
Other NLP tasks

- **Constituency parsing:** breaks the phrase into sub-phrases. Non-terminals in the tree are types of phrases, the terminals are the words in the sentence, and the edges are unlabeled
- **Dependency parsing:** connects the words according to their relationships. Each vertex in the tree represents a word, child nodes are words that are dependent on the parent, and edges are labeled by the relationship (Stanford dependencies for English here)

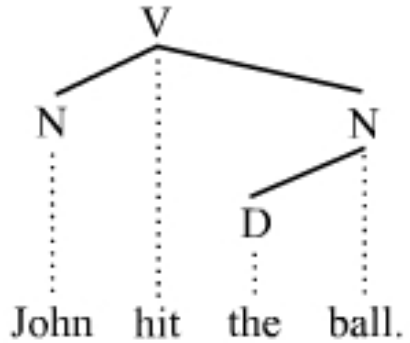
See http://stackoverflow.com/questions/10401076/difference-between-constituency-parser-and-dependency-parser#10401433 and http://nlp.stanford.edu:8080/parser/

Other NLP tasks

- There are other tasks in typical NLP pipelines
  - **Parsing:** Construct a tree that represents the syntactic structure of the string according to some grammars [1]



Constituency-based parse tree



Dependency-based parse tree

  - **Co-reference resolution:** finding all expressions that refer to the same entity in a text. e.g: “I voted for Nader because **he** was most aligned with **my** values”, **she** said
    - *He* refers to Nader; *My* refers to I; *She* refers to my

VU VRIJE UNIVERSITEIT AMSTERDAM

[1] Figures from Wikipedia