Natual Language Processing

Alessia Mondolo

August 13, 2019

Academy AI

Introduction to NLP

Introduction

Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language.

The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.

Most NLP techniques rely on machine learning to derive meaning from human languages.

NLP applications

Natural Language Processing is the driving force behind the following common applications:

- **Language translation** applications such as Google Translate.
- **Word Processors** such as Microsoft Word and Grammarly that employ NLP to check grammatical accuracy of texts, or programs able to perform automatic summarization, plagiarism detection, document classification (e.g. anti-spamming, sentiment polarity), etc.
- **Interactive Voice Response** (IVR) applications used in call centers to respond to certain users' requests.
- **Personal assistant** applications such as OK Google, Siri, Cortana, and Alexa.

Why is NLP difficult?

The rules that dictate the passing of information using natural languages are not easy for computers to understand:

- Some of these rules can be high-leveled and abstract; for example, when someone uses a sarcastic remark to pass information.
- On the other hand, some of these rules can be low-levelled; for example, using the character "s" to signify the plurality of items.

Comprehensively understanding the human language requires **understanding both the words and how the concepts are connected** to deliver the intended message.

While humans can easily master a language, the **ambiguity** and **imprecise** characteristics of the natural languages are what make NLP difficult for machines to implement.

NLP problems

- **World-knowledge:** representing world-knowledge is mandatory for understanding natual language (AI-completeness).
- **Multilinguality:** different languages require different models and resources. Also, there is the problem of the use of words from other languages.
- **Evaluation:** how to evaluate the correctness/suitability of a translation/summary?
- **Variability:** different sentences can refer to one meaning (e.g.: "Where can I get a map?", "I need a map", "need map" (non standard text)).
- **Ambiguity:** one sentence reefers to different meanings. E.g.: Ester said about Alice: "I made her duck", possible meanings:
  - I cooked waterfowl for her
  - I cooked the waterfowl she owned
  - I created the duck she owns
  - I caused her to quickly lower her head or body
  - I turned her into waterfowl

NLP techniques

Syntactic analysis and **semantic analysis** are the main techniques used to complete Natural Language Processing tasks:

- **Syntax:** it refers to the arrangement of words in a sentence such that they make grammatical sense. In NLP, syntactic analysis is used to **assess how the natural language aligns with the grammatical rules**.
- **Semantics:** it refers to the meaning that is conveyed by a text. Semantic analysis is one of the difficult aspects of Natural Language Processing that has not been fully resolved yet. It involves applying computer algorithms to **understand the meaning and interpretation of words and how sentences are structured**.

Syntax

Some of the most used techniques in syntax analysis are:

- **Sentence breaking:** it involves placing sentence boundaries on a large piece of text.
- **Word segmentation:** it involves dividing a large piece of continuous text into distinct units.
- **Lemmatization:** it entails reducing the various inflected forms of a word into a single form for easy analysis.
- **Morphological segmentation:** it involves dividing words into individual units called morphemes.
- **Part-of-speech tagging:** it involves identifying the part of speech for every word.
- **Parsing:** it involves undertaking grammatical analysis for the provided sentence.
- **Stemming:** it involves cutting the inflected words to their root form.

Semantics

Some of the most used techniques in semantic analysis are:

- **Named entity recognition (NER):** it involves determining the parts of a text that can be identified and categorized into preset groups. Examples of such groups include names of people and names of places.
- **Word sense disambiguation:** it involves giving meaning to a word based on the context.
- **Natural language generation:** it involves using databases to derive semantic intentions and convert them into human language.

Syntax Analysis

Words tokenization

Before a program can process natural language, we need identify the words that constitute a string of characters. This is important because the meaning of text generally depends on the relations of words in that text.

By default text on a computer is represented through string values. These values store a sequence of characters (nowadays mostly in UTF-8 format). The first step of an NLP pipeline is therefore to split the text into smaller units corresponding to the words of the language we are considering. In the context of NLP we often refer to these units as tokens, and the process of extracting these units is called tokenization.

Tokenization is considered boring by most, but it's hard to overemphasize its importance, seeing as it's the first step in a long pipeline of NLP processors, and if you get this step wrong, all further steps will suffer.

Words tokenization

Goal: split plain text into basic units.

Use: Information Retrieval (IR) tasks, text categorization, sentence splitting, language identification, text normalization, etc.

Different **basic units** depending on the task:

- **Naïve tokenizations:** split by blanks and punctuation marks occurring after alphanumeric string.
- **Complex tokenizations:** names, clitics (”’m”, ”’re”, ”’ve”, etc.), abbreviations, collocations (sequences of words that frequently occur together, e.g. ”break a leg”, ”nn the one hand”), etc.

Examples of tokenization

Blanks	outer punct.	Abbr.	Clitics	Colloc.	text normalized
Of	Of	Of	Of	Of_course	Of_course
course	course	course	course		
I'll	I'll	I'll	I	I	I
			'll	'll	will
go	go	go	go	go	go
to	to	to	to	to	to
U.P.C.	U.P.C	U.P.C	U.P.C	U.P.C	Universitat. . .
	.	.	.	.	.
	"	"	"	"	"
"Daily,	Daily	Daily	Daily	Daily	Daily
	,	,	,	,	,
Mr.	Mr	Mr.	Mr.	Mr.	Mister
	.				
John	John	John	John	John	John_Smith
Smith..."	Smith	Smith	Smith	Smith	
	...	...	...	...	...
	"	"	"	"	"

Sentence segmentation

Many NLP tools work on a sentence-by-sentence basis. The next preprocessing step is hence to segment streams of tokens into sentences. In most cases this is straightforward after tokenization, because we only need to split sentences at sentence-ending punctuation tokens.

Sentence segmentation

Goal: Recognition of sentence boundaries in plain text (e.g., '.' '?' '!' '...').

Type of task:

- Language-dependent task (e.g. German: "Mein 2. Semester kommt bald zu Ende."; traditional chinese?)
- Domain-dependent task
  (e.g. "It is expressed as (x=1)? T.add('-') : T.add(x).")

Methods:

- Hand-crafted rules
- Machine learning methods

Problems of sentence segmentation

Main problems:

- Abbreviations and acronyms (most difficult one): "I will meet with **Mr. Smith** to talk about it.", "Lisa run **25 km. She** ended up in **N.Y.**"
- Ellipsis: "There're different methods (A, B, **. . .**) but **. . .** "
- Internal quotation: "'Stop!' he shouted."
- Ordinal numbers (German)
- Special cases: "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, etc.
- Regular expressions for general cases, abbreviations, ellipsis, etc.:

Ex: / ([?!])+/ $\rightarrow$ t $\in$ s_boundary
Ex: / (\.){3} [A-Z]/ $\rightarrow$ t $\in$ s_boundary
Ex: / [?!.] \) [A-Z]/ $\rightarrow$ t $\in$ s_boundary
Ex: / ($TITLE) \. / $\rightarrow$ t $\notin$ s_boundary
Ex: / [A-Z] \. / $\rightarrow$ t $\notin$ s_boundary

Problem: highly expensive adaptation to new languages (rules and abbreviation classes).

Supervised ML for sentence splitting

- The most frequently used: EM (Expectation-Maximization), SVM (support Vector Machine), CRF (Coditional Random Fields), etc.
- 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_X: 1srt token before '.' is X

tok-2_X: 2nd token before '.' is X

tok+1_X: 1st token after '.' is X

len_tok-1_X: length of 1st token before '.' is X

len_tok-2_X: length of 2nd token before '.' is X

len_tok+1_X: length of 1st token after '.' is 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_X: abbreviation class of 1st token before '.' is 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	e^-	tok-1_Mr
	len_tok-1_3		len_tok-1_2
	CAP_tok-1		up_tok-1
	tok-2_to		tok-2,
	len_tok-2_2		len_tok-2_1
	lo_tok-2		class_tok-1_title
	tok+1_"		tok+1_John
	len_tok+1_1		len_tok+1_4
			up_tok+1

Unsupervised ML for sentence splitting

- Based on corpus statistics
- Easily adaptable to new languages: They require large unnanotated 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]

NLTK

NLTK

The Natural Language Toolkit (NLTK) is an open source Python library for Natural Language Processing.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

Installing NLTK: link
Natural Language Processing with Python: link book

Today's homework

Exercise

Notebook