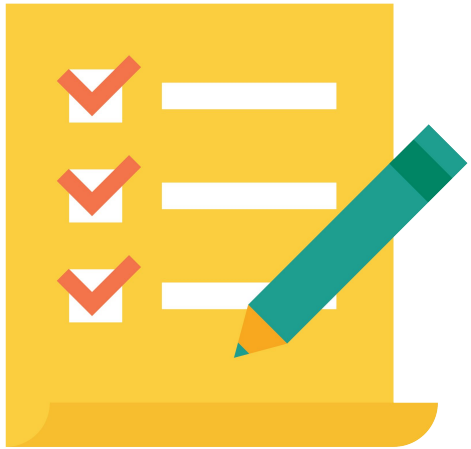
# Behavior-driven development
# An introduction

# THE “ACCEPTANCE” PROBLEM

teal.blue
digital solutions

## What is the *“Definition of Done”* for a project feature? How to test whether it is satisfied?



- How to avoid ambiguity and miscommunications?
- Is a qualitative description in a high level document sufficient?

- How to test the “right things”?
- Are unit tests sufficient?

- What is essential, what is “nice to have”?

# THE “LANGUAGE” PROBLEM

teal.blue
digital solutions

## How can all stakeholders in a digital project agree on the *Definition of Done*?


- Clients, project managers, designers, developers, testers:

  different ways to describe problems and solutions

- **Clients**: business business needs
- **Project managers**: high level specifications
- **Designers**: user journeys, mockups
- **Developers**: unit tests, api specifications
- **Testers**: end-to-end tests

# ALL STAKEHOLDERS CARE ABOUT VALUE

teal.blue digital solutions

**The need for a feature could be captured by a *story* that describes system behavior as *value* to the user.**

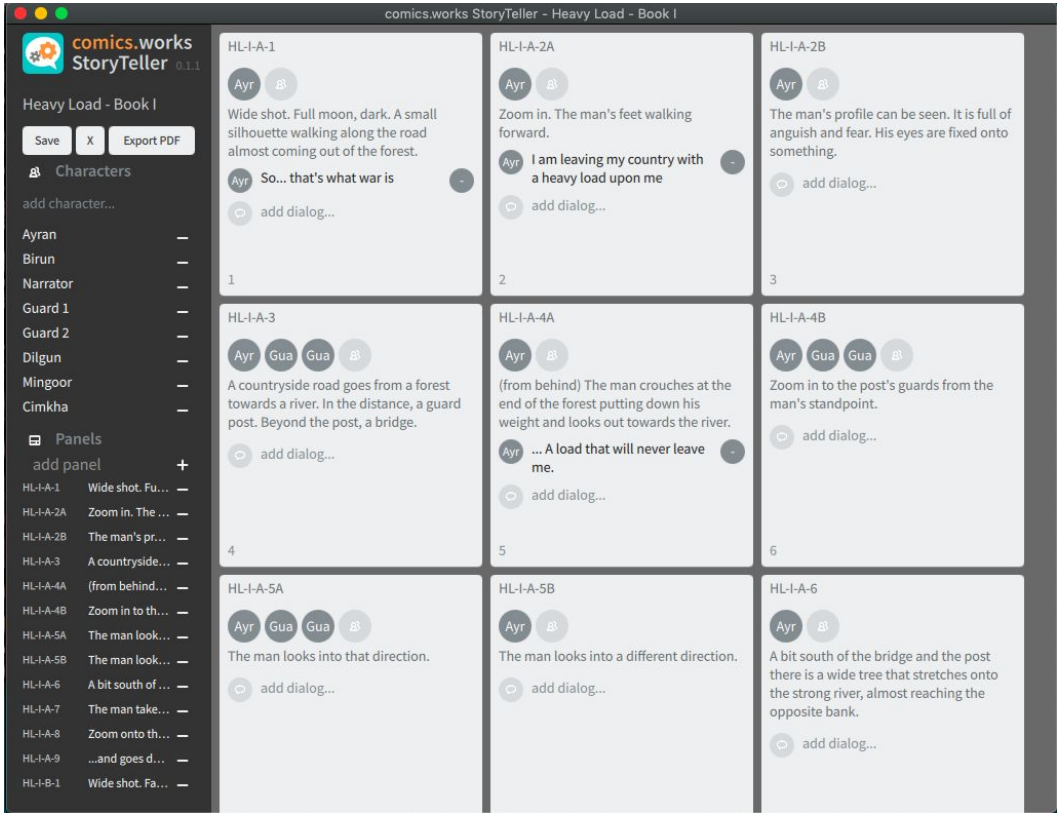
```
As a [X]
I want [Y]
so that [Z]
```

- Y is some feature
- Z is the benefit or value of the feature
- X is the person (or role) who will benefit

If the system fulfills a story's *acceptance criteria*, it's behaving correctly; if it doesn't, it isn't.
Acceptance criteria define how value is provided to users.

# EXAMPLE: COMICS WORKS STUDIO (DEMO)

teal.blue digital solutions




```
Feature: create project

As a comic creator
I want to create a project
so that I can keep all
     related work and settings
     in one place
```

# DEFINING ACCEPTANCE CRITERIA

teal.blue digital solutions

**Acceptance criteria should be provided in a *language* that is common among all project stakeholders.**

Behavior-driven development relies on such a language:

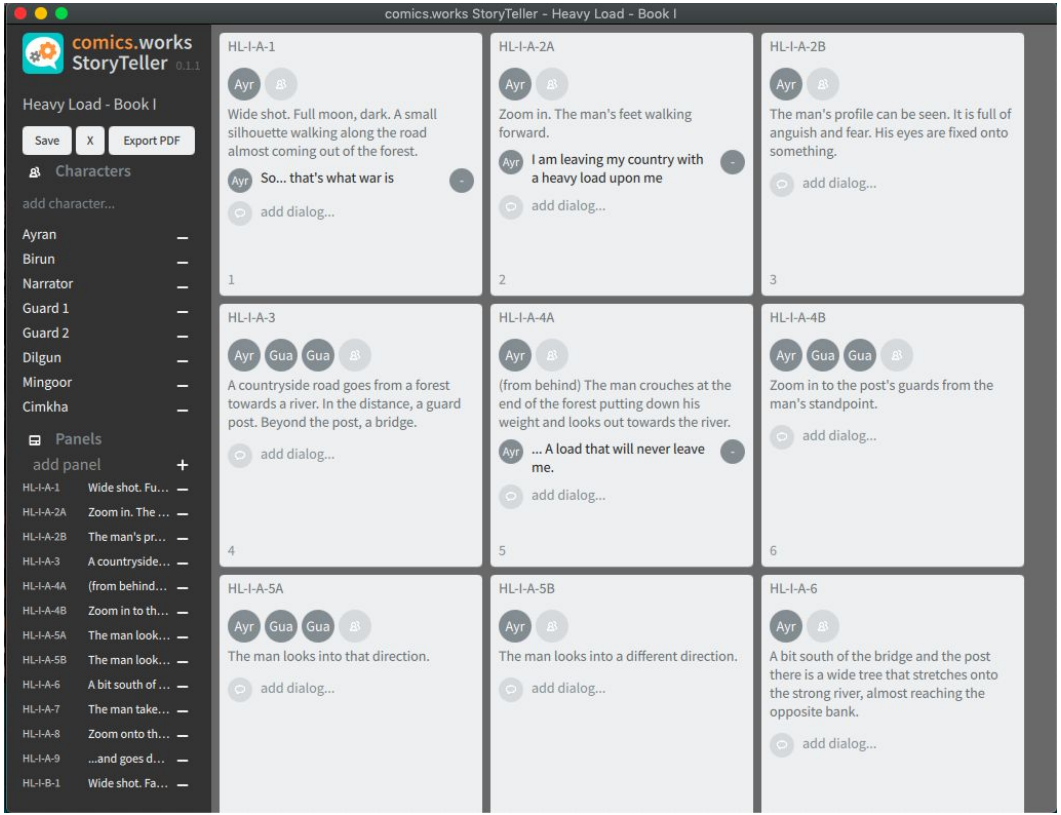
controlled *natural language* (human language)

```
Given [A]
When [B]
Then [C]
```

- A is some initial *context* (the givens)
- B is an *event* that occurs
- C are the *outcomes* that the feature must ensure

# EXAMPLE: COMICS WORKS STUDIO (DEMO)

teal.blue digital solutions




```
Feature: describe panel
As a comic creator
I want to describe a panel
so that I can give it a meaning
 even before filling it

Scenario: completed
Given the panel with id "panel1"
 is created
When I try to describe panel
 "panel1" as "my first panel"
Then a description is added to
 panel "panel1"
And the description for panel
"panel1" reads "my first panel"
```

This *controlled natural language* is called Gherkin

# PRECONDITIONS DETERMINE BEHAVIOR

teal.blue
digital solutions

## Typically we want to ensure that, for a given story, from different preconditions we get different outcomes.

Each set of Given-When-Then's is called a *scenario*.

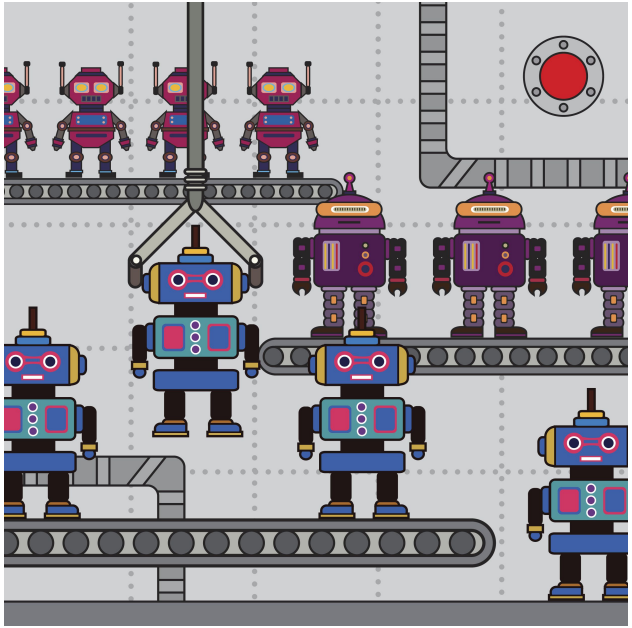
```
Feature: create panel

  Scenario: completed
    Given no panel with id "panel1" exists in the current project
    When I try to create a panel with id "panel1"
    Then the panel with id "panel1" is created
    And I can lookup the panel with id "panel1" in the current project
```

```
  Scenario: abandoned - panel already exists
    Given a panel with id "panel1" exists in the current project
    When I try to create a panel with id "panel1"
    Then the panel with id "panel1" is not created
    And I am told that a panel with id "panel1" already exists
```

# TESTING ACCEPTANCE CRITERIA

teal.blue digital solutions

## BDD acceptance criteria testing can be automated!



- Create a “world” (a mini application) to store all and only the code objects needed for a scenario.
- Pass the world to each of the “givens” so they can populate the world with known state (preconditions).
- Tell the event to “occur in” the world. The event carries out the actual behaviour in the scenario.
- Check any outcomes that were defined for the story.

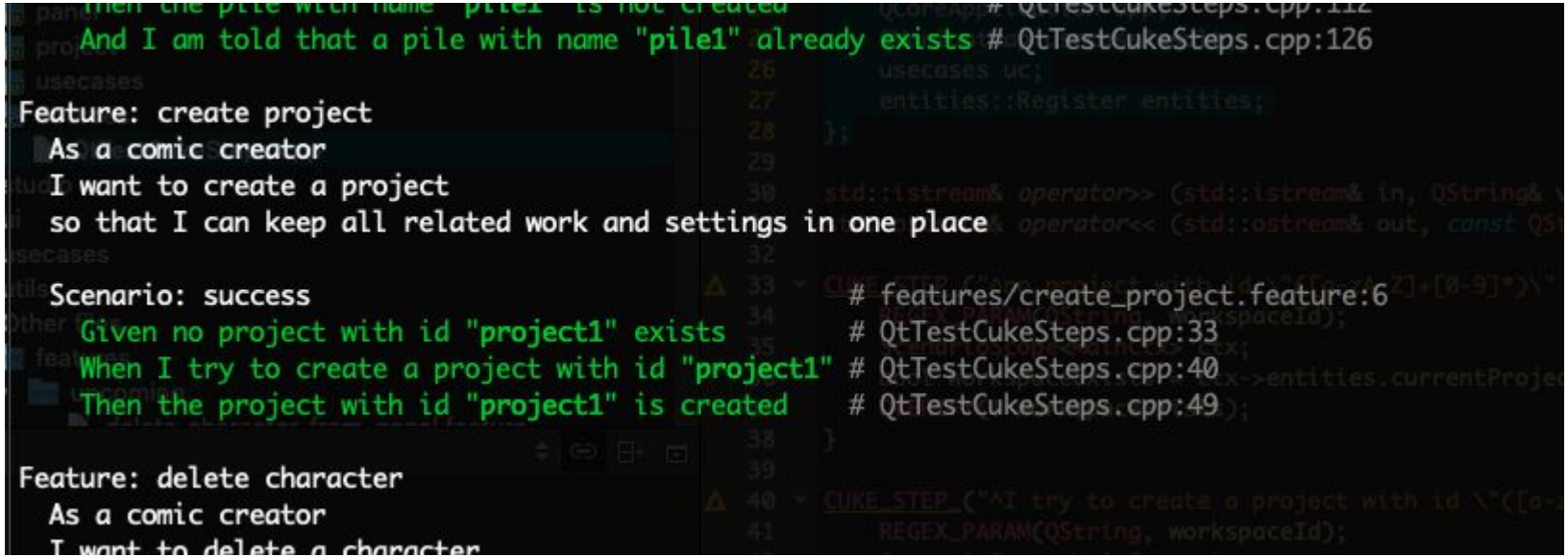
# EXAMPLE: COMICS WORKS STUDIO (DEMO)

teal.blue
digital solutions

```
GIVEN("^no project with id \"([a-zA-Z]+[0-9]*)\" exists$") {
    REGEX_PARAM(QString, workspaceId);
    ScenarioScope<MainCtx> ctx;
    bool workspaceExists = ctx->entities.currentProject != nullptr;
    QVERIFY(! workspaceExists);
}

WHEN("^I try to create a project with id \"([a-zA-Z]+[0-9]*)\"$") {
    REGEX_PARAM(QString, workspaceId);
    ScenarioScope<MainCtx> ctx;
    ...
    ctx->uc.create_project(workspaceId);
    ...
    ctx->usecaseResult = usecaseResult.takeFirst().at(0).toMap();
}

THEN("^the project with id \"([a-zA-Z]+[0-9]*)\" is created$") {
    REGEX_PARAM(QString, workspaceId);
    ScenarioScope<MainCtx> ctx;
    QCOMPARE(ctx->entities.currentProject->eid(), workspaceId);
}
```

# EXAMPLE: COMICS WORKS STUDIO (DEMO)

teal.blue
digital solutions



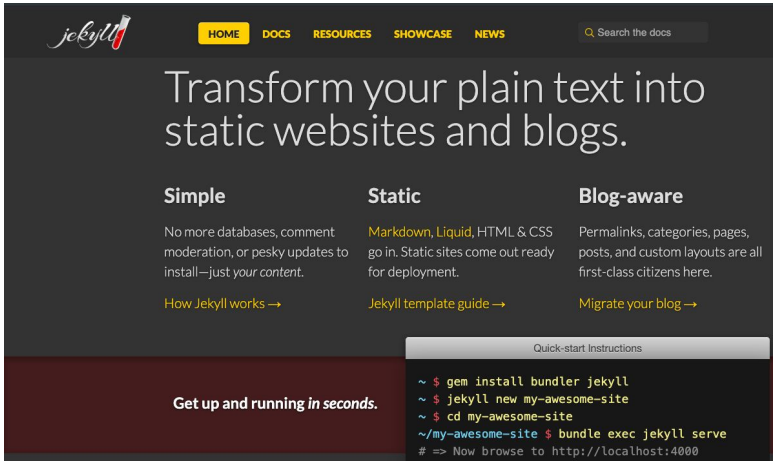
# SOME OPEN SOURCE PROJECTS USING BDD

teal.blue
digital solutions

## Jekyll

a blog-aware static site generator in Ruby

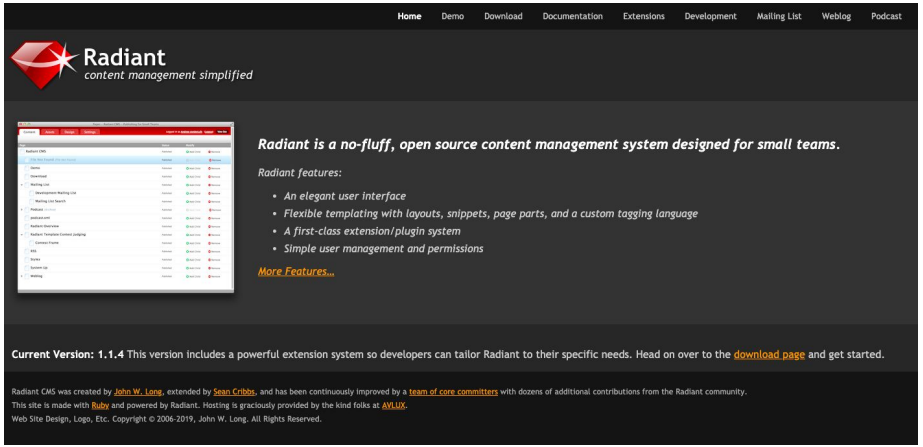
https://jekyllrb.com




## RadiantCMS

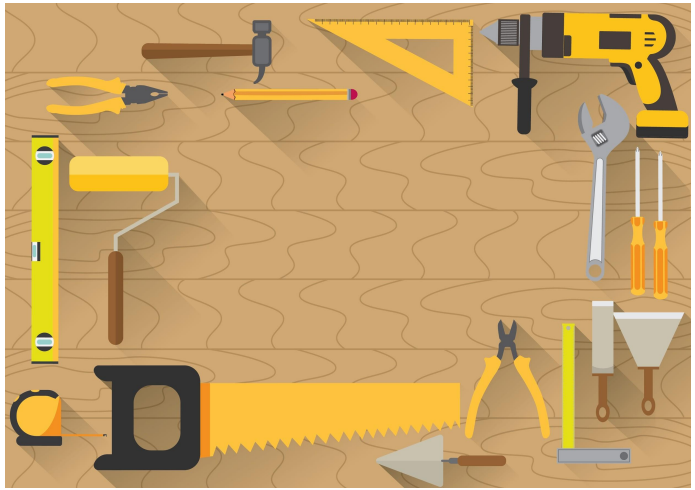
an open source content management system designed for small teams

http://radiantcms.org



# POPULAR BDD TOOLS & FRAMEWORKS

teal.blue
digital solutions



https://dannorth.net/introducing-bdd/

cucumber https://cucumber.io

Open Source - Supports many programming languages

**Gherkin**: the formalized language used in BDD
https://cucumber.io/docs/gherkin/reference/

SMARTBEAR
HipTest https://hiptest.com

Cloud-based continuous testing environment with code generation - free for Open Source projects

jbehave https://jbehave.org

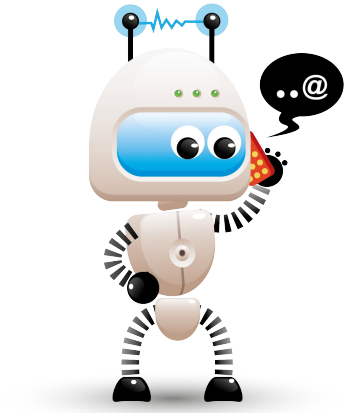
From BDD's creator Dan North


# This year's project

## NaturalAPI: a toolkit to narrow the gap between project specifications and APIs.

Contact point: marco.piccolino@teal.blue

teal.blue
digital solutions

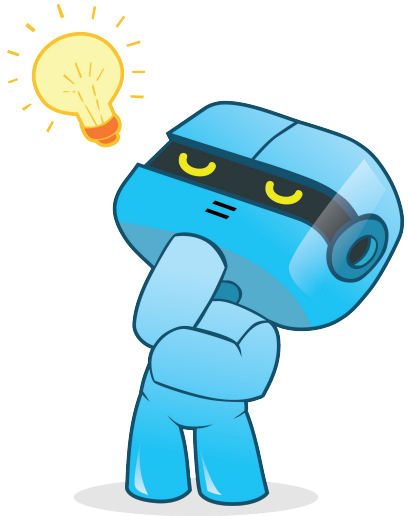


# Natural Language Processing
# A brief introduction

# THE MAIN TASK

teal.blue
digital solutions

## How to program computers that can analyze and generate natural language data?



"Language is highly **ambiguous**– it relies on subtle cues and contexts to convey meaning.

Take this simple example: "I love flying planes."

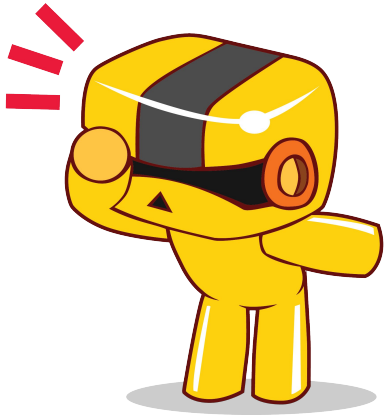
Do I enjoy participating in the act of piloting an aircraft? Or am I expressing an appreciation for man-made vehicles engaged in movement through the air on wings?"

*Source: Vincent Chen*

# THE GROUND PROBLEM

teal.blue
digital solutions

## Current computers do not like ambiguity.
## Either it's a 0, or it's a 1.



Ambiguity in language is present at various levels, e.g.:

- **Pragmatics** (deal with context contributing to meaning)
  "Wow, what a beautiful hat!" (said with a disgusted face)

- **Semantics** (deals with sentence meaning)
  "We saw her duck" (duck name vs. duck verb)

- **Syntax** (deals with sentence structure)
  "John ate the cookies on the couch" (John or cookies on couch?)

- **Phonetics** (deals with language sounds)
  "ice cream" vs "I scream"

# APPROACHES TO SOLVE THE ISSUE

teal.blue digital solutions

## Historically, two main, complementary approaches: *Symbolic* and *Statistical* Natural Language Processing

**Symbolic**

define rules to parse/generate language “by hand”,
often using formalisms from classical computer science.

Example: hand-written context-free grammars (CFGs) to perform syntactic analysis (parsing).

**Statistical**

infer rules (or rule probabilities) from (usually large) data collections by so-called *machine learning* methods.

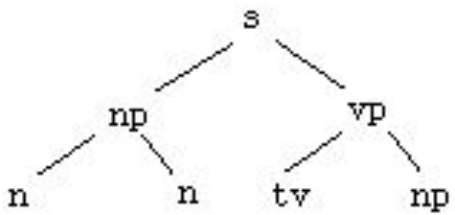
Example: neural networks to perform text categorization (is this a legal text or a piece of literature?).

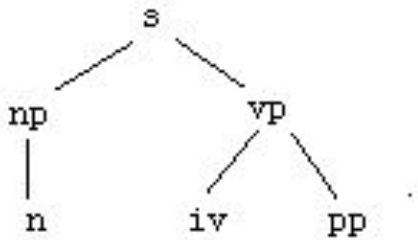
# EX.: SYMBOLIC NLP - CFG PARSING

teal.blue digital solutions

Figure 2. An ambiguous grammar and partial parse trees for "fruit flies like an apple."

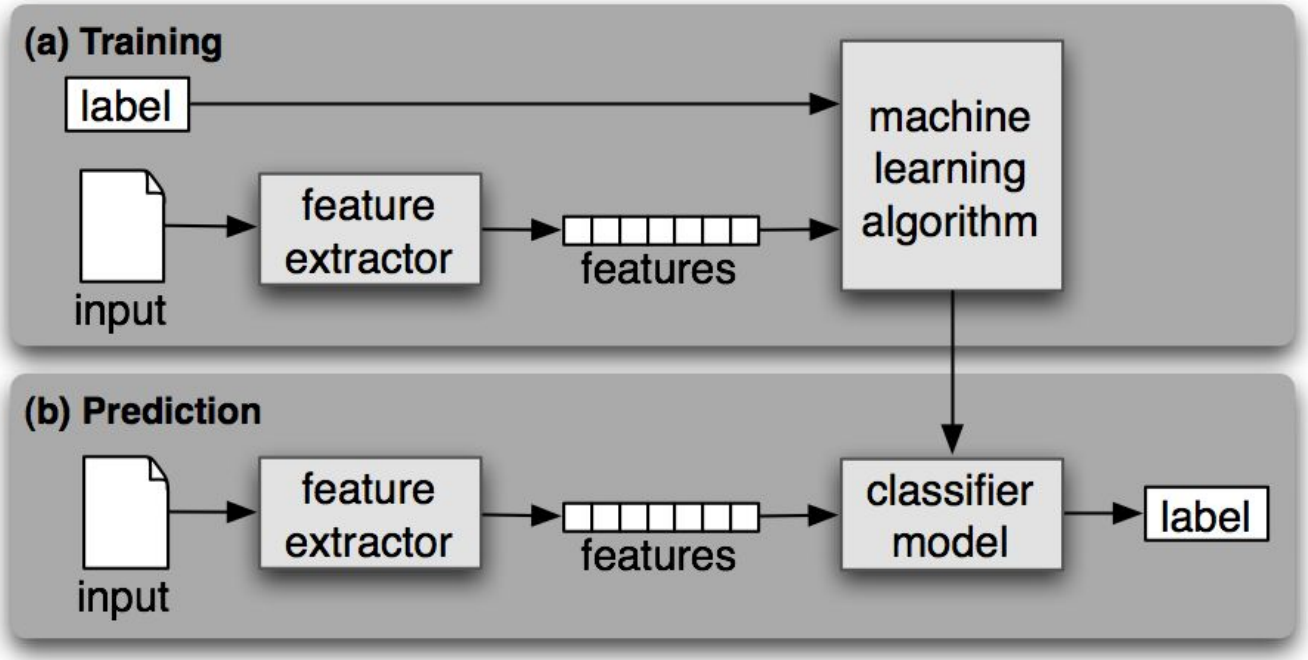
```
s    ⟶ np vp
np   ⟶ det n
     ⟶ n
     ⟶ n n
vp   ⟶ tv np
     ⟶ iv pp
pp   ⟶ prep np
det  ⟶ a
     ⟶ an
n    ⟶ fruit
     ⟶ apple
     ⟶ flies
iv   ⟶ flies
tv   ⟶ like
prep ⟶ like
```




Source: http://www.bowdoin.edu/~allen/nlp/nlp1.html

# EX.: STAT. NLP - TEXT CATEGORIZATION




Source: https://www.nltk.org/book/ch06.html

# PROBABILITY

teal.blue digital solutions

**Most NLP systems nowadays make use of probability to deal with language ambiguity at various levels.**

| | | | | |
|---|---|---|---|---|
| S | ⇒ | NP | VP | 1.0 |
| VP | ⇒ | Vi | | 0.4 |
| VP | ⇒ | Vt | NP | 0.4 |
| VP | ⇒ | VP | PP | 0.2 |
| NP | ⇒ | DT | NN | 0.3 |
| NP | ⇒ | NP | PP | 0.7 |
| PP | ⇒ | P | NP | 1.0 |

| | | | |
|---|---|---|---|
| Vi | ⇒ | sleeps | 1.0 |
| Vt | ⇒ | saw | 1.0 |
| NN | ⇒ | man | 0.7 |
| NN | ⇒ | woman | 0.2 |
| NN | ⇒ | telescope | 0.1 |
| DT | ⇒ | the | 1.0 |
| IN | ⇒ | with | 0.5 |
| IN | ⇒ | in | 0.5 |

- Probability of a tree $t$ with rules

$$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$$

is $p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$ where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$.

Source: Michael Collins

# USEFUL NLP TASK: Lemmatization

teal.blue
digital solutions

**Remove inflectional endings to return the base dictionary form of a word (lemma).**

| | original_word | lemmatized_word |
|---|---|---|
| 0 | trouble | trouble |
| 1 | troubling | trouble |
| 2 | troubled | trouble |
| 3 | troubles | trouble |

| | original_word | lemmatized_word |
|---|---|---|
| 0 | goose | goose |
| 1 | geese | goose |

Source: https://kavita-ganesan.com/

# USEFUL NLP TASK: Part-of-speech tagging

teal.blue
digital solutions

**Mark up a word in a text as corresponding to a particular part of speech.**

The_DT first_JJ time_NN he_PRP was_VBD shot_VBN in_IN the_DT hand_NN as_IN he_PRP chased_VBD the_DT robbers_NNS outside_RB ...

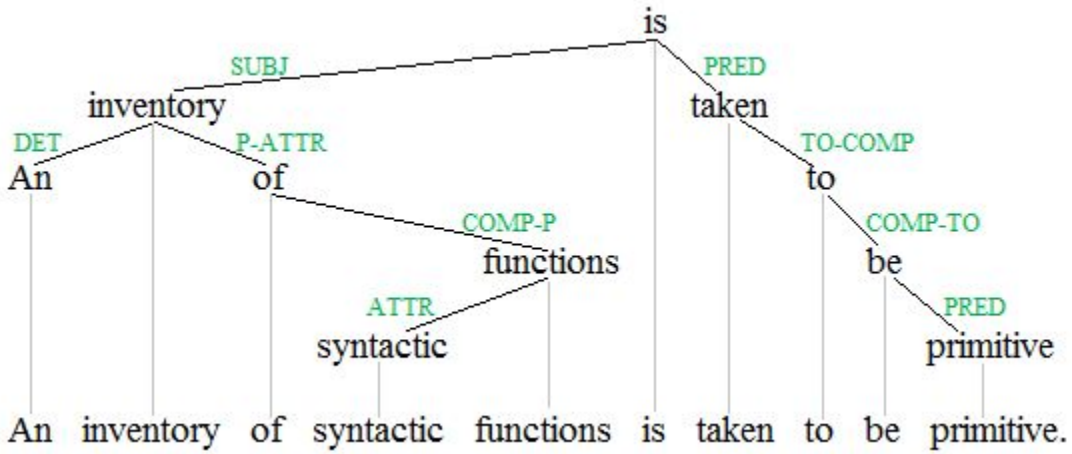
| first | time | shot | in | hand | as | chased | outside |
|---|---|---|---|---|---|---|---|
| JJ | NN | NN | IN | NN | IN | JJ | IN |
| RB | VB | VBD | RB | VB | RB | VBD | JJ |
| | | VBN | RP | | | VBN | NN |
| | | | | | | | RB |

Source: Lluís Màrquez

# USEFUL NLP TASK: Dependency parsing

teal.blue digital solutions

**Determine sentence structure by the relation between a word (a head) and its dependents.**

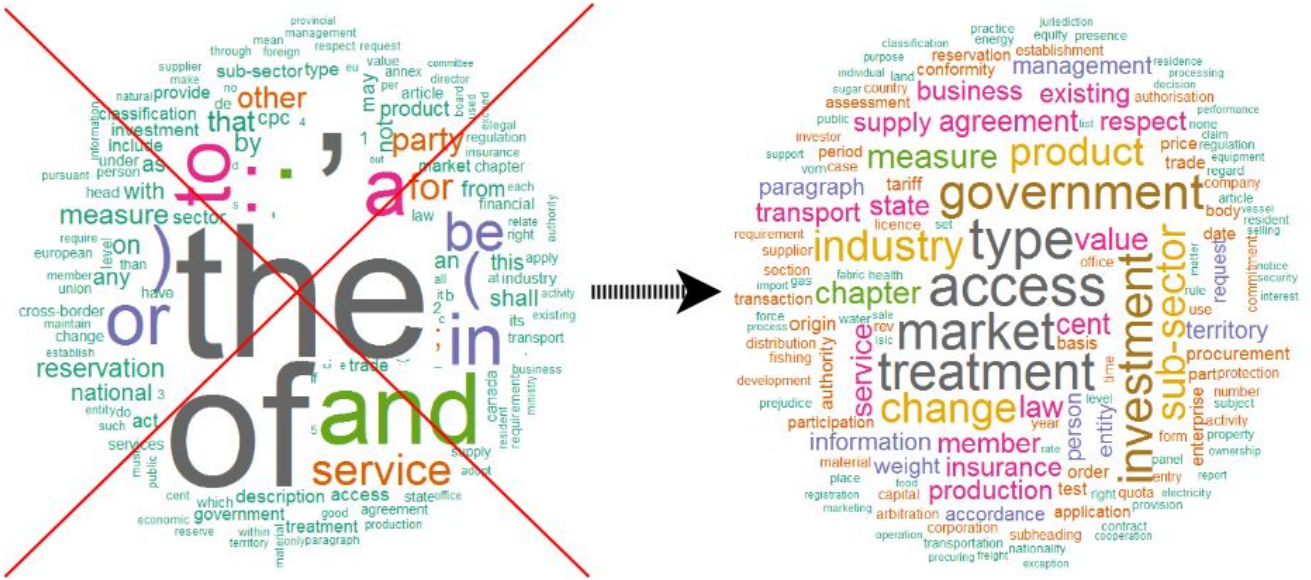



Source: Wikipedia

# USEFUL NLP TASK: Terminology extraction

teal.blue digital solutions

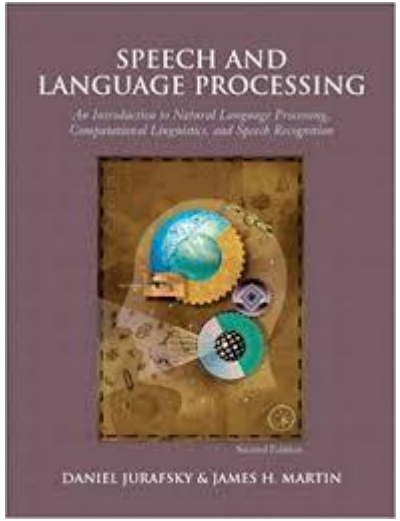
**Automatically extract relevant terms (or keywords) from a given collection of texts (a corpus).**




Source: https://www.r-bloggers.com/

# NLP TOOLS & RESOURCES

teal.blue
digital solutions




**Natural language processing**
https://en.wikipedia.org/wiki/Natural_language_processing
An overview of NLP's subdisciplines and tasks

**Speech and Language Processing** (3rd ed. Draft - free!)
Dan Jurafsky and James H. Martin
https://web.stanford.edu/~jurafsky/slp3/
Great introduction to NLP and speech technology

**Dependency parsing - Stanford parser**
https://nlp.stanford.edu/software/nndep.html
Brief definition of dependency parsing, state-of-the art parser.

**SpaCy**
https://spacy.io/
Open-source software library for advanced NLP


# This year's project

## NaturalAPI: a toolkit to narrow the gap between project specifications and APIs.

Contact point: marco.piccolino@teal.blue