

*Natural language  
processing with Python*

**Hi all!**

Iván Compañy  
@pyivanc

# Index

1. Introduction
  1. NLP
  2. MyTaste
2. Theory
  1. Data processing
    1. Normalizing
    2. Tokenizing
    3. Tagging
  2. Data post-processing
    1. Classification
    2. Word count
3. Practice

## 1.1. NLP

***Natural language processing (NLP)** is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages*

*Wikipedia*

***Natural language processing (NLP)** is all about human text processing*

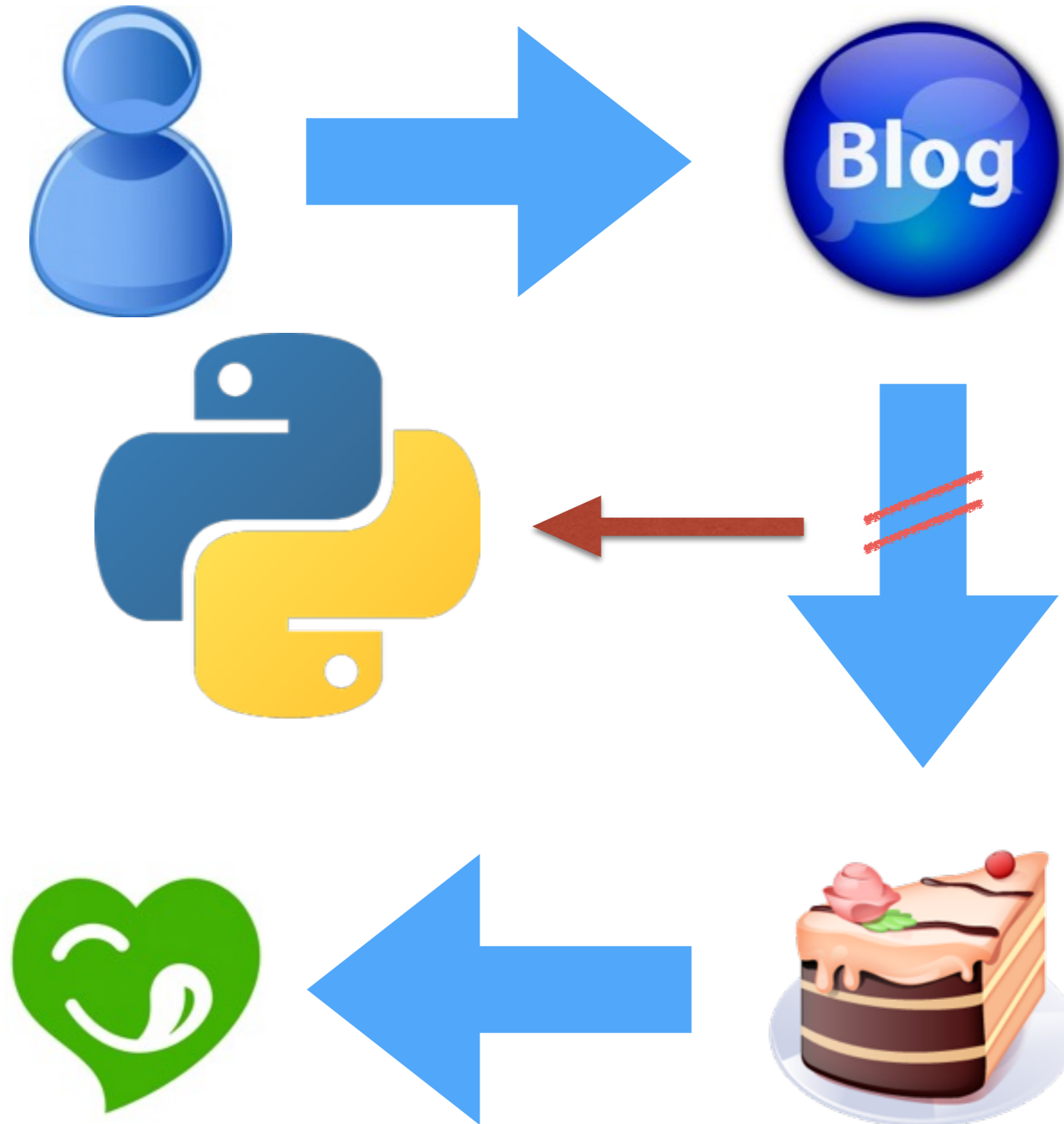
*Me*

## 1.2. MyTaste

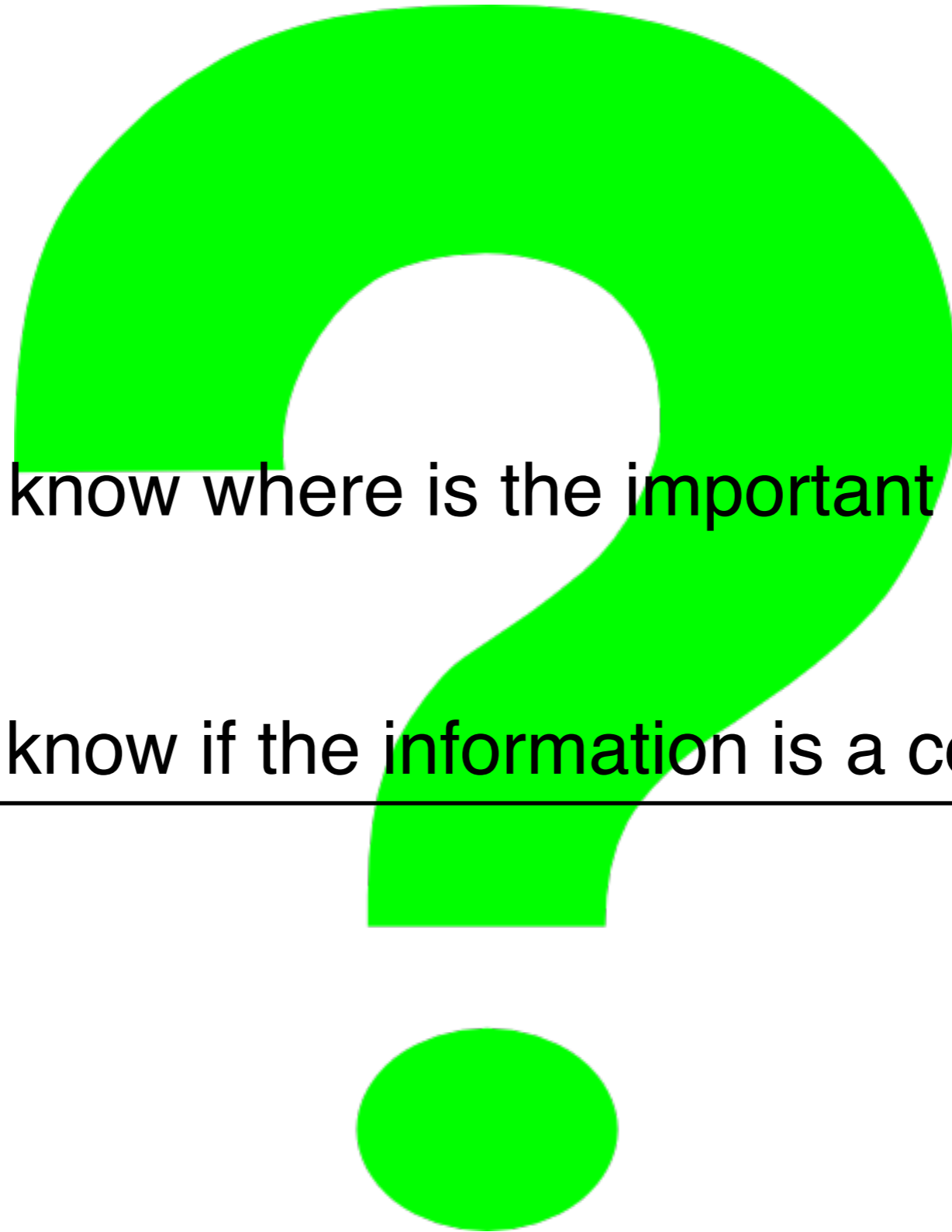
*myTaste*

<http://www.mytaste.com/>

## 1.2. MyTaste



## 1.2. MyTaste

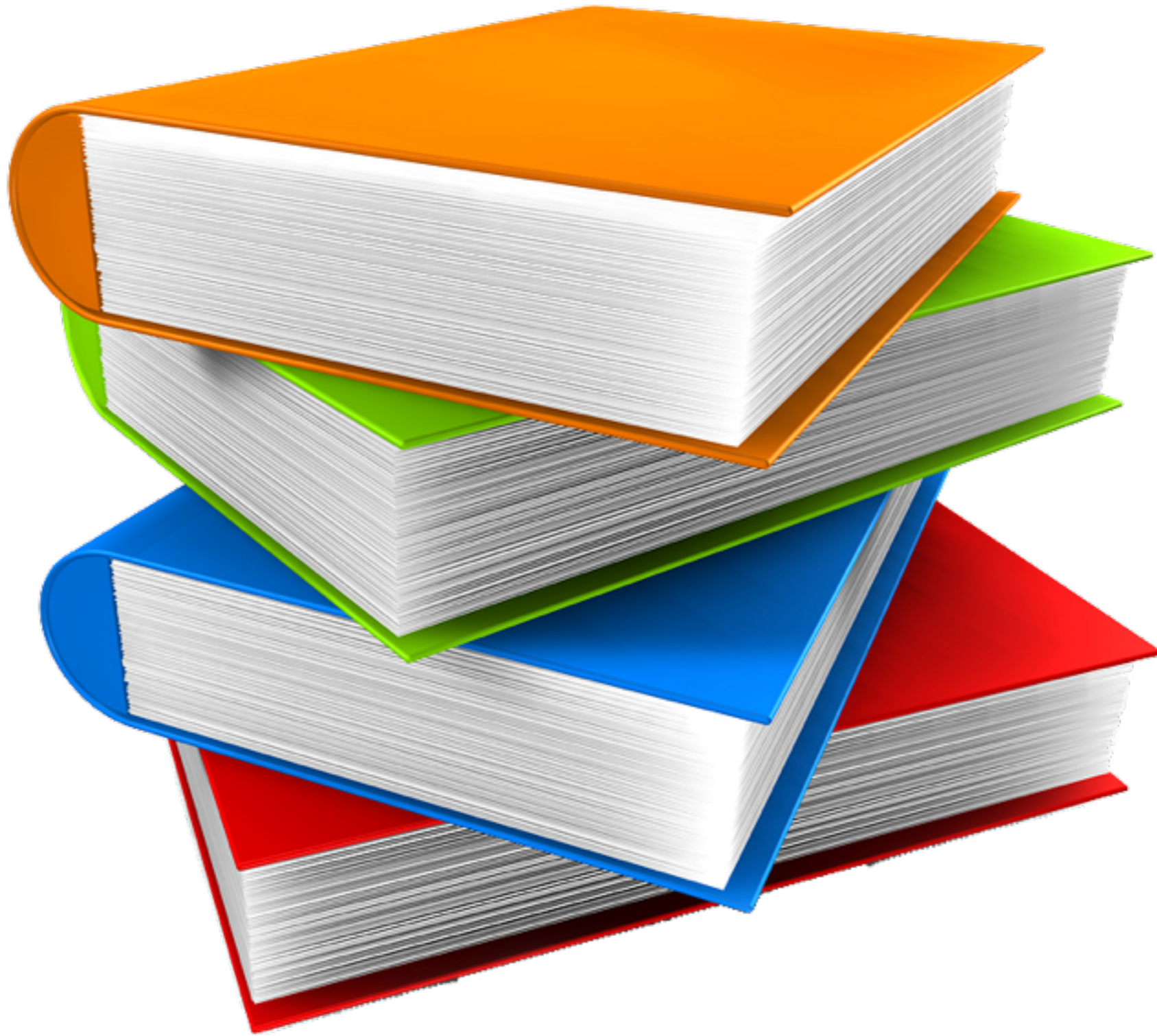


1. How do I know where is the important information?

2. How do I know if the information is a cooking recipe?

---

## 2. Theory





## 2.1. Data processing

yo soy programador python

DET VRB N N

1. Normalize: Adapt the text depending of your needs
2. Tokenize: Split text in identifiable linguistic units
3. Tagging: Classify words into their parts-of-speech

#### Content selection

##### Recipes

I love the chocolate

Everybody loves the chocolate

I still love chocolate!

Why you don't love chocolate?

My GF is chocolate

##### Non Recipes

Once upon a time..

Remember, remember...

```
print("hello Python")
```

Fly, you fools!

Let there be rock

### Feature selection

- ♦ *“A recipe must contain the word ‘chocolate’”*
- ♦ *“A recipe must contain at least one verb”*

**What!**

Training

**MACHINE LEARNING ALGORITHM**

## 2.2. Data post-processing

### 1. Classification

Test input

This is my recipe of chocolate  
This is the beginning of anything  
I love chocolate  
Why you don't love chocolate?  
I'm becoming insane  
...

Machine  
learning  
algorithm

This is my recipe of chocolate → **recipe**  
This is the beginning of anything → **non recipe**  
I love chocolate → **recipe**  
Why you don't love chocolate? → **recipe**  
I'm becoming insane → **non recipe**

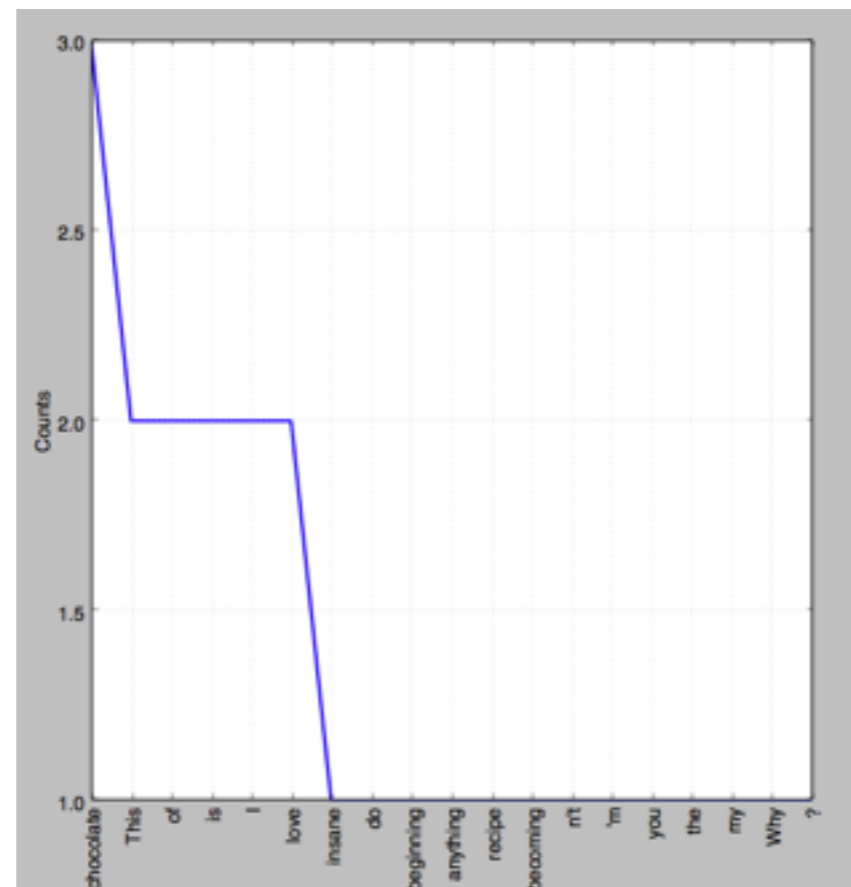
## 2.2. Data post-processing

## 2. Word count

Input

This is my recipe of chocolate  
This is the beginning of anything  
I love chocolate  
Why you don't love chocolate?  
I'm becoming insane  
...

Frequency  
distribution  
algorithm





# NLTK

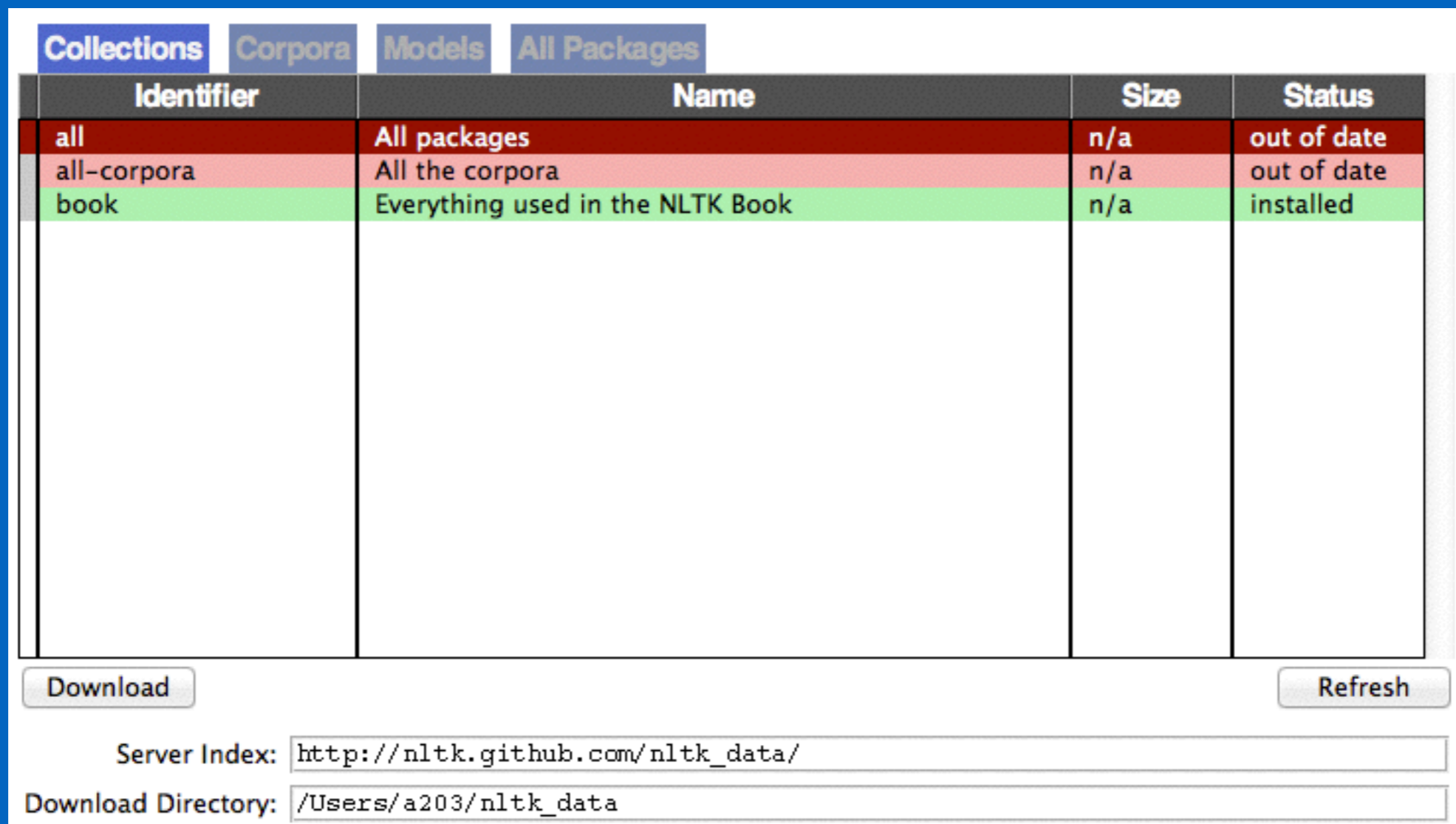
<http://www.nltk.org/>

Installing:

```
pip install nltk
```

Downloading corpora:

```
python -c "import nltk;nltk.download()"
```



The screenshot shows the NLTK data download interface. At the top, there are four tabs: "Collections", "Corpora", "Models", and "All Packages". Below the tabs is a table with the following columns: "Identifier", "Name", "Size", and "Status". The table contains three rows of data:

Identifier	Name	Size	Status
all	All packages	n/a	out of date
all-corpora	All the corpora	n/a	out of date
book	Everything used in the NLTK Book	n/a	installed

Below the table, there are two buttons: "Download" and "Refresh". At the bottom, there are two input fields: "Server Index:" with the value `http://nltk.github.com/nltk_data/` and "Download Directory:" with the value `/Users/a203/nltk_data`.



# Data processing

```
import nltk

#Sample text
text = "I am a Python programmer"

#Normalize (the 'I' character in english must be in uppercase)
text = text.capitalize()

#Tokenize
text_tokenized = nltk.word_tokenize(text)

#We create the tagger
from nltk.corpus import brown

default_tag = nltk.DefaultTagger('UNK')
tagger = nltk.UnigramTagger(brown.tagged_sents(), backoff=default_tag)

#Tagging
text_tagged = tagger.tag(text_tokenized)

#Output
#[('I', u'PPSS'), ('am', u'BEM'), ('a', u'AT'), ('python', u'NN'), ('programmer', u'NN')]
```

# Data post-processing: Classification

```
from nltk.corpus import brown

with open('tagged_recipes.txt', 'rb') as f:
    recipes = []
    for recipe in f.readlines():
        recipes.append(recipe, 'recipe')

non_recipes = []
for line in brown.tagged_sents()[ :1000]:
    non_recipes.append((line, 'non_recipe'))

all_texts = (recipes + non_recipes)

random.shuffle(all_texts)
```

# Data post-processing: Classification

```
def feature(text):  
    has_chocolate = any(['chocolate' in word.lower() for (word, pos) in text])  
    is_american = any(['america' in word.lower() for (word, pos) in text])  
    return {'has_chocolate': has_chocolate, 'is_american': is_american}  
  
featuresets = [(feature(text), t) for (text, t) in all_sents]  
  
train_set, test_set = featuresets[:18000], featuresets[18000:]  
  
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

# Data post-processing: Classification

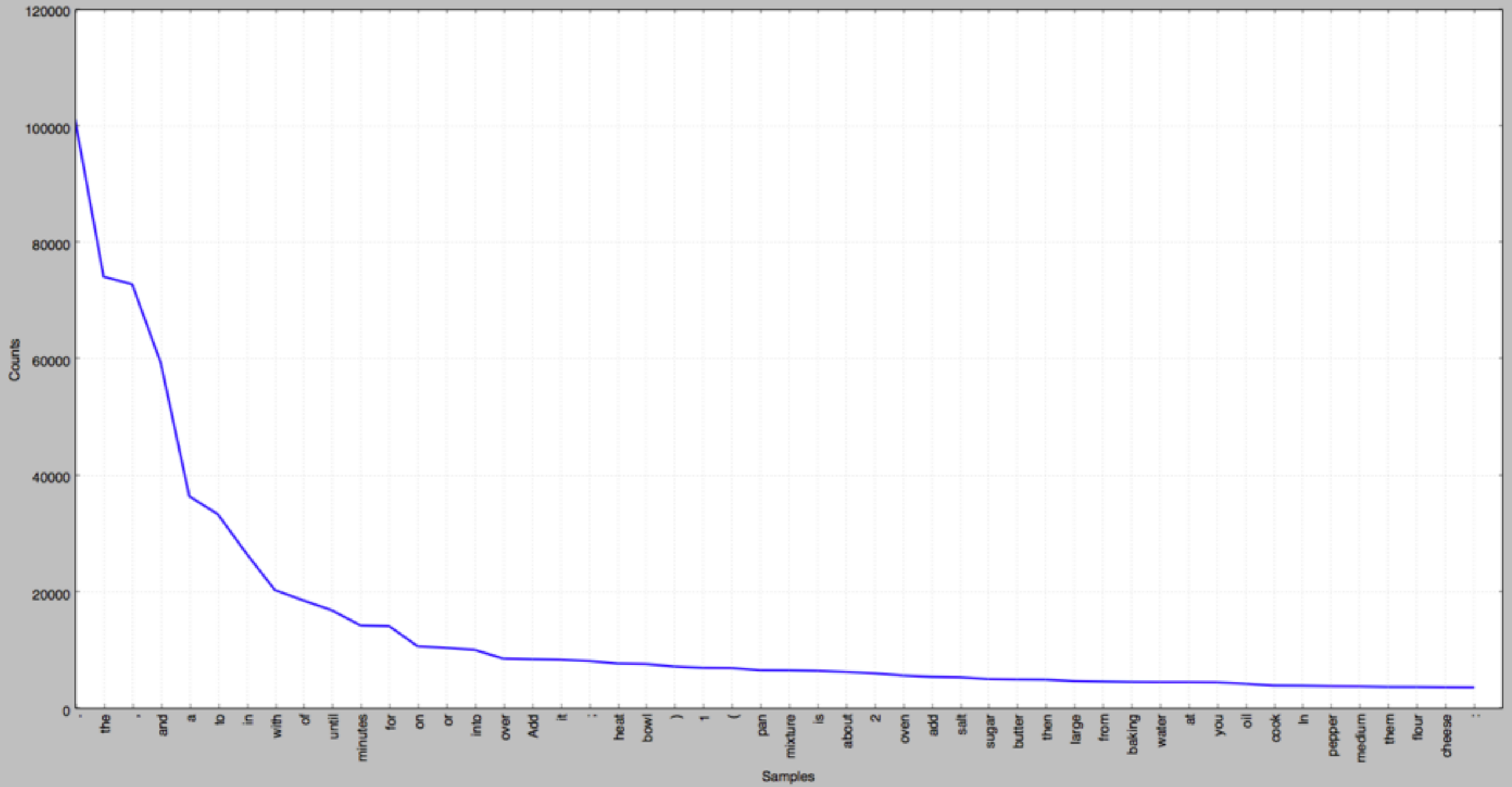
```
print classifier.show_most_informative_features(5)
"""
Most Informative Features
      has_chocolate = True           recipe : non_re =      86.6 : 1.0
        is_american = True           non_re : recipe =     70.4 : 1.0
        is_american = False          recipe : non_re =      1.0 : 1.0
      has_chocolate = False          non_re : recipe =      1.0 : 1.0
"""
print nltk.classify.accuracy(classifier, test_set)
"""
0.514
"""
```

# Data post-processing: Word count

```
import nltk

with open("recipes.txt") as f:
    tokenized_text = nltk.word_tokenize(f.read())
    fdist = nltk.FreqDist(tokenized_text)
    fdist.plot(50)
```

# Data post-processing: Classification



## Examples of usage:

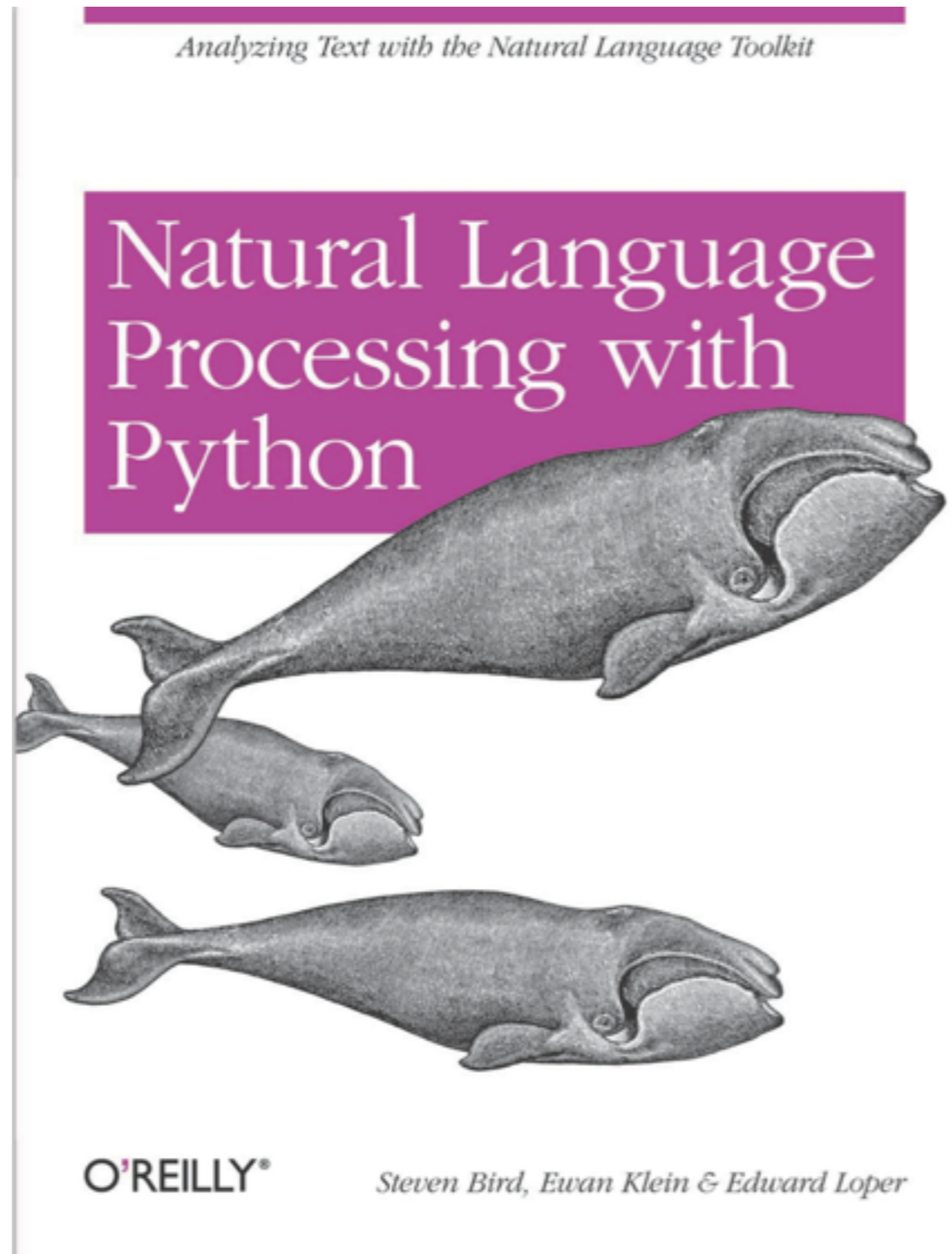
- Text recognition
- Tag detection in websites
- Market analysis
- Ads system
- Translators
- ...

## More things I didn't explain:

- Analyze group of texts (Chunk tagging)
- Semantic analysis
- How to create a corpus
- ...



## More info:



# Thank you!

Iván Compañy  
@pyivanc