

# Logische Dokumentenanalyse - Arbeiten mit OCR-Dokumenten / Techniken / Grundlagen

---

Andreas Neumann M.A.  
Centrum für Informations- und Sprachverarbeitung

# Themen

---

1. Physischen Aufbau eines OCR-Dokuments als Objekt darstellen
2. Positionelle Relationen zwischen Objektelementen als Eigenschaften modellieren
3. Globale Beobachtungen zur Extraktion fundamentaler Layouteinheiten nutzen

Physischen Aufbau eines OCR-Dokuments als  
Objekt darstellen

# Physischer Dokumentenaufbau

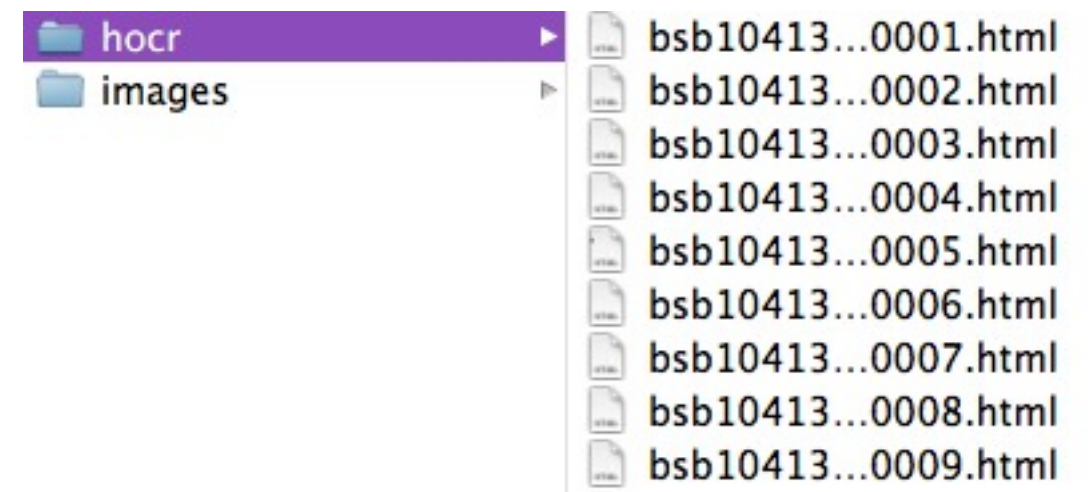
---

- Ein Dokument besteht aus Seiten
- Seiten bestehen aus Blöcken
- Blöcke bestehen aus Paragraphen mit Zeilen
- Zeilen bestehen aus Wörtern
- Wörter bestehen aus Zeichen

# Ein OCR-Dokument - Abbild der Seiten im Dateiensystem

---

- Seiten sind innerhalb eines Dokuments indirekt und direkt kodiert:
  - Jede Seite entspricht einer Datei
  - Innerhalb der Datei (kann) die Seitennummer als Attribut der Seite kodiert sein



# Eine OCR-Datei - Weitere Elemente innerhalb einer Seite

---

- Innerhalb einer Seite sind kodiert:

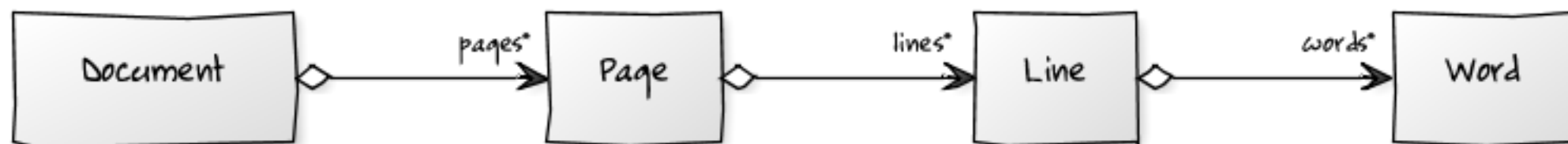
- Block
- (Paragraph)
- Zeile
- Wort (direkt oder indirekt durch Character)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>OCR Output</title>
<meta http-equiv='content-type' content='text/html; charset=utf-8' />
<meta http-equiv='content-style-type' content='text/css' />
<meta name='ocr-capabilities' content='ocr_page ocr_par ocrx_word ocr_line' />
<meta name='ocr-system' content='ABBY fre-8.0.1.1024' />
<meta name='ocr-number-of-pages' content='1' />
</head><body bgcolor='#ffffff'>
<div class='ocr_page' title='bbox 0 0 1326 1326;ppageno 1'>
<div class='ocrx_block' title='bbox 0 0 1320 2088'>
<p class='ocr_par' title='bbox 0 0 1320 2088'
style='font-size:0pt;font-family:"Arial";font-style:normal'></p>
</div>
<div class='ocrx_block' title='bbox 614 490 685 520'>
<p class='ocr_par' title='bbox 614 490 685 520'
style='font-size:10pt;font-family:"Arial";font-style:normal'><span class='ocr_line'
title='bbox 614 490 685 520'><span class='ocrx_word' title='bbox 614 490 685
520'>^,,"</span></span></p>
```

# Vereinfachter Aufbau des Document Object

---

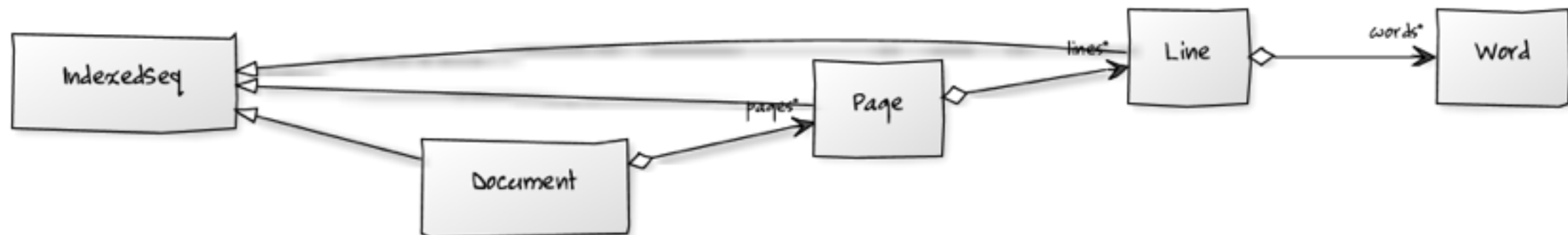
- Ein Dokument wird zur Analyse in ein Dokumentobjekt eingelesen
- Jedes Dokument hat Seiten mit Zeilen
- Jede Zeile besteht aus Wörtern



# Elemente als Indexierte Sequenzen

---

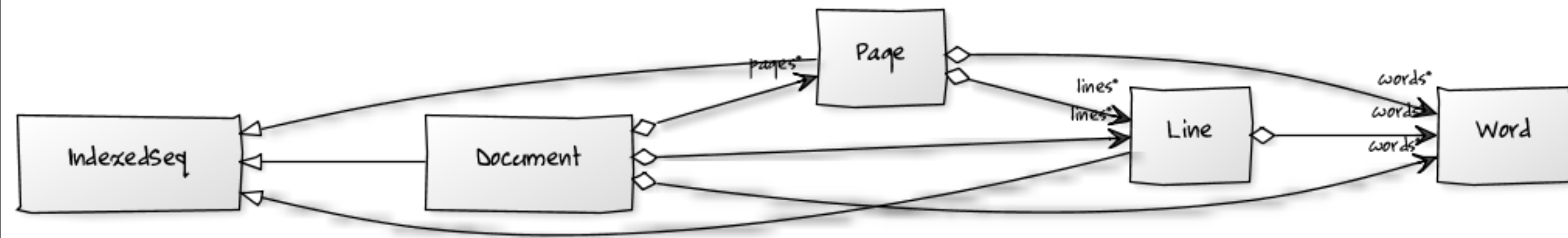
- Erlaubt auf einzelne Elemente gezielt zuzugreifen
- Ordnung wichtig





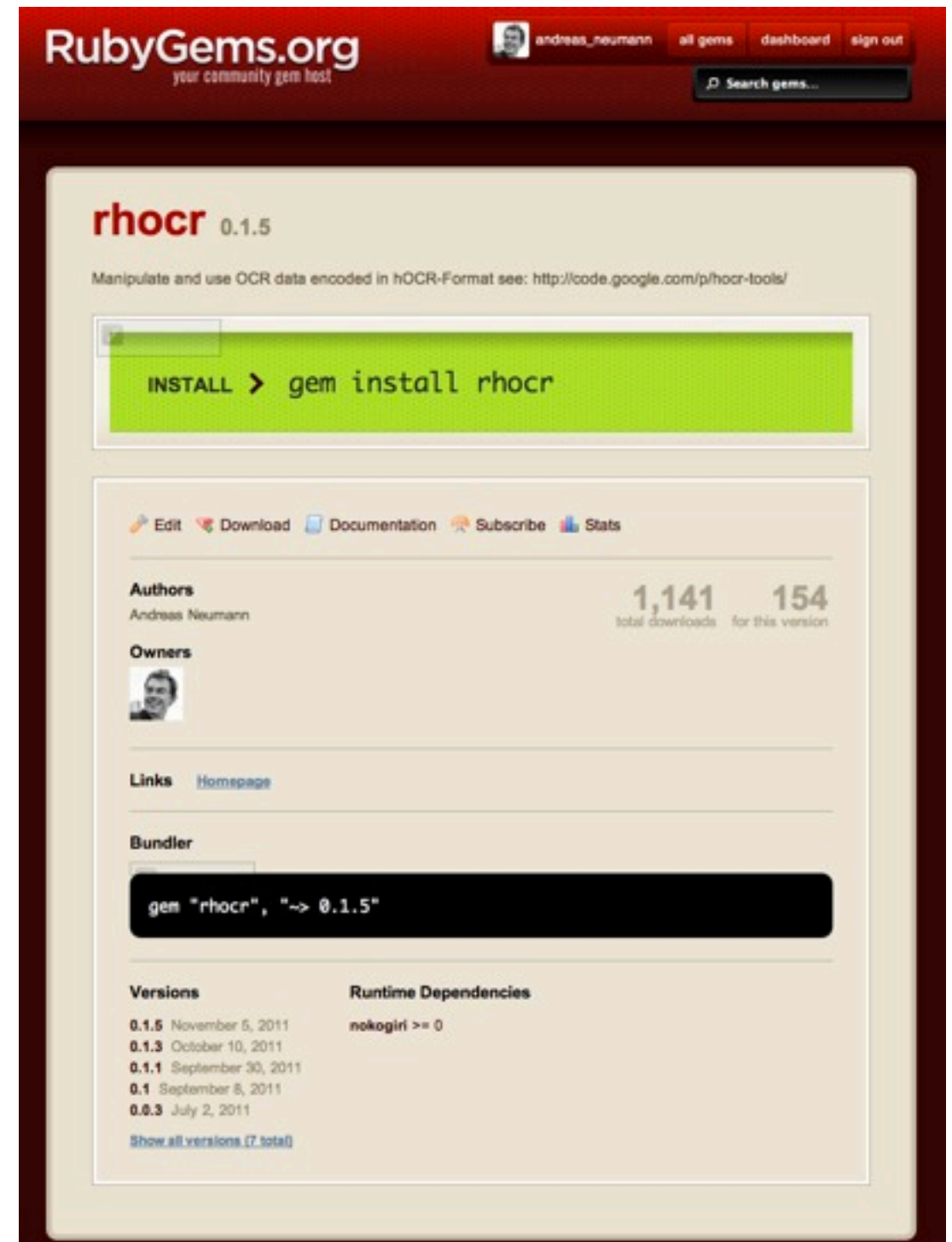
# Erweiterter Aufbau

- Jedes Dokument besteht auch aus Zeilen und Wörtern
- Jede Seite besteht auch aus Wörtern



# Eine mögliche Implementierung für hOCR (rhOCR)

- rhOCR ist eine Ruby-Bibliothek zum bearbeiten von hOCR-Dokumenten
- Installation: `gem install rhocr`
- <https://rubygems.org/gems/rhocr>
- <https://github.com/daandi/rhocr>



The screenshot shows the RubyGems.org page for the **rhocr** gem, version 0.1.5. The page has a red header with the RubyGems.org logo and navigation links. The main content area is white with a green bar for the installation command. Below this, there are sections for authors, owners, links, bundler, versions, and runtime dependencies.

**RubyGems.org** your community gem host

andreas\_neumann all gems dashboard sign out

Search gems...

## rhocr 0.1.5

Manipulate and use OCR data encoded in hOCR-Format see: <http://code.google.com/p/hocr-tools/>

**INSTALL >** `gem install rhocr`

Edit Download Documentation Subscribe Stats

**Authors**  
Andreas Neumann

**Owners**  
1,141 total downloads 154 for this version

**Links** Homepage

**Bundler**  
`gem "rhocr", "~> 0.1.5"`

**Versions**

0.1.5	November 5, 2011
0.1.3	October 10, 2011
0.1.1	September 30, 2011
0.1	September 8, 2011
0.0.3	July 2, 2011

[Show all versions \(7 total\)](#)

**Runtime Dependencies**  
nokogiri >= 0

# rhOCR - Document

---

- Methoden um Dateien und Verzeichnisse hinzuzufügen
- Methoden zum durchlaufen von Seiten, Zeilen und Wörter des Dokuments

## Methods

```
1  ::new
2  #add_file
3  #add_files
4  #add_image_to_page
5  #add_page
6  #add_pages
7  #each_line
8  #each_page
9  #each_word
10 #page
```

# rhOCR - Document - Methoden zum hinzufügen von Seiten

---

```
describe 'methods to #add_page an access pages' do

  before(:each) do
    @test_document = OCRDocument.new
    @test_document.add_page 'data/test.html'
    @test_document.add_page 'data/Seite_Tagebuch_H_C_Lang_08.html'
    @test_document.add_page 'data/Seite_Die_Gartenlaube_242.html'
  end

  it 'should have #pages' do
    @test_document.pages.keys.sort.should == [20, 33, 242]
  end

  it 'should retrieve page by pagenumber' do
    page = @test_document.pages[33]
    page_words = []
    page.each_word do |word|
      page_words << word
    end
    page_words.length.should == 346
  end

  it 'should have an alternate syntax for accessing pages by #page' do
    @test_document.page(105).should == @test_document.pages[105]
  end

  it 'has a method to add an image to a file #add_image_to_page' do
    @test_document.add_image_to_page(20, "path")
    @test_document.page(20).image.should == "path"
  end

end
```

# rhOCR - Document - Methoden zum iterieren über Seitenelemente

---

```
describe 'Methods to iterate over words, lines and pages' do

  before(:each) do
    @test_document = OCRDocument.new
    @test_document.add_page 'data/test.html'
    @test_document.add_page 'data/Seite_Tagebuch_H_C_Lang_08.html'
    @test_document.add_page 'data/Seite_Die_Gartenlaube_242.html'
  end

  it 'should have a method to iterate over document lines #each_line' do
    a = []
    @test_document.each_line do |line|
      a << line
    end
    a.size.should == 237
  end

  it 'should have a method to iterate over document words #each_word' do
    a = []
    @test_document.each_word do |word|
      a << word
    end
    a.size.should == 2071
  end

  it 'should have a method to iterate pages #each_page' do
    start_number = 0
    @test_document.each_page do |page|
      page.page_number.should > start_number
      start_number = page.page_number
    end
  end
end
```

# rhOCR - Document - Aufbau

```
#coding: utf-8

require_relative 'ocr_page'

class OCRDocument
  attr_reader :pages, :page_count

  def initialize
    @pages = Hash.new()
    @page_count = 0
  end

  def add_pages( list_o_pages )
    raise "no files given" if list_o_pages.empty?
    list_o_pages.each do |file|
      add_page(file)
    end
  end

  def add_page( file )
    page = OCRPage.new( file )
    @pages[page.page_number] = page
    @page_count += 1
  end

  def add_image_to_page(page_number, image_path)
    @pages[page_number].image = image_path
  end

  def page( number )
    @pages[number]
  end
end
```

```
def each_page
  sorted_pages = @pages.keys.sort
  sorted_pages.each do |page_key|
    yield @pages[page_key]
  end
end

def each_line
  for page in @pages.values do
    page.each_line do |line|
      yield line
    end
  end
end

def each_word
  for page in @pages.values do
    page.each_line do |line|
      line.each do |word|
        yield word
      end
    end
  end
end

alias :add_files :add_pages
alias :add_file :add_page
end
```

# Beispiel alternative Bibliothek - (hOCR - Scala; in Entwicklung)

---

- Das fünfte Wort der vierten Zeile auf der sechsten Seite des Dokuments
- Alle Zeilen des Dokuments
- Alle Wörter des Dokuments
- Verschachtelte Elemente des Dokuments durchlaufen.

```
val doc =
  Document.fromFolder(getClass.getResource(
    source("/testdoc/hocr/").getFile)

doc.pages(5).lines(3).words(4)
doc(5)(3)(4)
doc.getPage(6).lines(3)(4)

doc.lines

doc.words

doc foreach {page =>
  page foreach {line =>
    line foreach {word =>
      //... do something
    }
  }
}
```

# Kurzusammenfassung

---

- Wir haben das Dokument als eine geordnete Menge von Seiten, welche eine geordnete Menge Zeilen und eine geordnete Menge Wörter enthalten dargestellt
- Durch den hier skizzierten Aufbau ist es außerdem möglich ein Dokument als eine geordnete Menge von Zielen oder eine geordnete Menge von Wörtern zu betrachten



Positionelle Relationen zwischen einzelnen  
Dokumentelementen als Objekteigenschaften  
definieren

# BoundingBox

---

- Jedes Seitenelement wird durch eine BoundingBox (siehe letzte Vorlesung) begrenzt
- Ein Seitenelement wird über zwei Punkte aufgespannt
- Diese Boxen können in unterschiedlichen Relationen zueinander stehen:
  - Sie können einander includieren ( Ein Wort wird von einer Zeile umschlossen)
  - Sie können link bzw. Rechts von einem Element liegen
  - Sie können über bzw. untereinander liegen
  - Sie können einen definierten Abstand zueinander haben
  - ...

# BoundingBox-Implementierung aus rhOCR

---

- Komplette Implementierung: [https://github.com/daandi/rhocr/blob/master/lib/hocr\\_box.rb](https://github.com/daandi/rhocr/blob/master/lib/hocr_box.rb)
- Implementierung in Auszügen auf der nächsten Folie

```
#coding: utf-8
```

```
class HOcrBox
```

```
    attr_reader :left, :top, :right, :width, :height, :bottom, :coordinates
```

```
    def initialize(* coordinates)
```

```
        @left, @top, @right, @bottom = coordinates.flatten.collect { |x| x.to_i}
```

```
        @height = @bottom - @top
```

```
        @width = @right - @left
```

```
        @coordinates = [ @left, @top, @right, @bottom ]
```

```
        if left > right || top > bottom then
```

```
            raise " Negative dimensions of OCRBox ar not allowed. left #{@left} / right #{@right} - top  
#{@top} / bottom #{@bottom}"
```

```
        end
```

```
    end
```

```
    def encloses?(other)
```

```
        @left <= other.left and
```

```
        @right >= other.right and
```

```
        @top <= other.top and
```

```
        @bottom >= other.bottom
```

```
    end
```

```
    def enclosed_by?(other)
```

```
        return other.encloses? self
```

```
    end
```

```
    ...
```

```
    def left_of?(other)
```

```
        @right < other.left
```

```
    end
```

```
end
```

# BoundingBox erstellen und Koordinaten abfragen

---

```
before(:each) do
  @box ||= HOCRBox.new(1, 2, 20, 8)
end

describe '#coordinates' do
  it 'should have coordinates' do
    @box.coordinates.should == [1, 2, 20, 8]
  end
  it 'should have #left' do
    @box.left.should == 1
  end
  it 'should have #right' do
    @box.right.should == 20
  end
  it 'should have #top' do
    @box.top.should == 2
  end
  it 'should have #bottom' do
    @box.bottom.should == 8
  end
  it 'should have height' do
    @box.height == 7
  end
  it 'should have width' do
    @box.width.should == 19
  end
end

end
```

# BoundingBox Relationen

---

```
describe '#enclosed_by?(element)' do
  it 'should be enclosed by Boxes bigger than itself' do
    @box.enclosed_by?( HOCRBox.new(0,1,21,9) ).should be_true
  end
  it 'should not be enclosed by Boxes smaller than itself' do
    @box.enclosed_by?( HOCRBox.new(2,3,19,7) ).should be_false
  end
  it 'should be enclosed by Boxes of the same size' do
    @box.enclosed_by?( @box ).should be_true
  end
end

...

describe '#left_of?(element)' do
  it 'should be left of any box-element that has a larger x1 than the current box has x2' do
    @box.left_of?( HOCRBox.new(21,2,21,8) ).should be_true #x1 == x2 -> Eine Linie
  end
  it 'should not be left of' do
    @box.left_of?( HOCRBox.new(1,2,2,8) ).should be_false
  end
end

describe '#left_distance_to(element)' do
  it 'element should be 5px left box' do
    HOCRBox.new(25,0,30,0).left_distance_to(@box).should == 5
  end
end

...

describe '#top_distance_to(element)' do
  it 'box should be 9px below of element' do
    HOCRBox.new(109,241,206,274).top_distance_to(HOCRBox.new(160,196,1117,232)).should == 9
  end
end

...
```

# BoundingBox-Implementierung in aus hOCR (in Entwicklung)

---

- Als Trait
- Erlaubt Mixin-Composition
- Es folgt die komplette Implementierung

```

package com.an_it.ocr

/**
 * AN-iT
 * Andreas Neumann
 * andreas.neumann@an-it.com
 * http://www.an-it.com
 */

trait BoundingBox {
  val coordinates: ((Int, Int), (Int, Int))
  lazy val ((left, top), (right, bottom)) = coordinates // has to be lazy!
  lazy val height = bottom - top
  lazy val width = right - left

  def encloses(other: BoundingBox ) = {
    left <= other.left &&
    right >= other.right &&
    top <= other.top &&
    bottom >= other.bottom
  }

  def enclosed_by(other: BoundingBox) = other.encloses(this)

  def leftOf(other: BoundingBox) = right < other.left
  def rightOf(other: BoundingBox) = left < other.right
  def leftDistanceTo(other: BoundingBox) = left - other.right
  def rightDistanceTo(other: BoundingBox) = other.leftDistanceTo(this)
  def topDistanceTo(other: BoundingBox) = top - other.bottom
  def bottomDistanceTo(other: BoundingBox) = other.topDistanceTo(this)

  def coordinatesToString = coordinates.toString()

  def toCSS(zoom: Double = 1) = "position:absolute; top:" + (top * zoom).toInt + "px; left:" +
(left * zoom).toInt + "px; height:" + (height * zoom).toInt + "px; width:" + (width * zoom).toInt
+ "px;"

}

```



# BoundingBox Scala - Benutzung

---

- Beispiele für die Nutzung folgen auf der nächsten Folie
- Kompletter Test unter: [https://github.com/daandi/hOCR/blob/master/src/test/scala/com/an\\_it/ocr/BoundingBoxSpec.scala](https://github.com/daandi/hOCR/blob/master/src/test/scala/com/an_it/ocr/BoundingBoxSpec.scala)

```

/**
 * AN-iT
 * Andreas Neumann
 * andreas.neumann@an-it.com
 * http://www.an-it.com
 */

import org.specs2.mutable.Specification
import org.specs2.specification.Scope

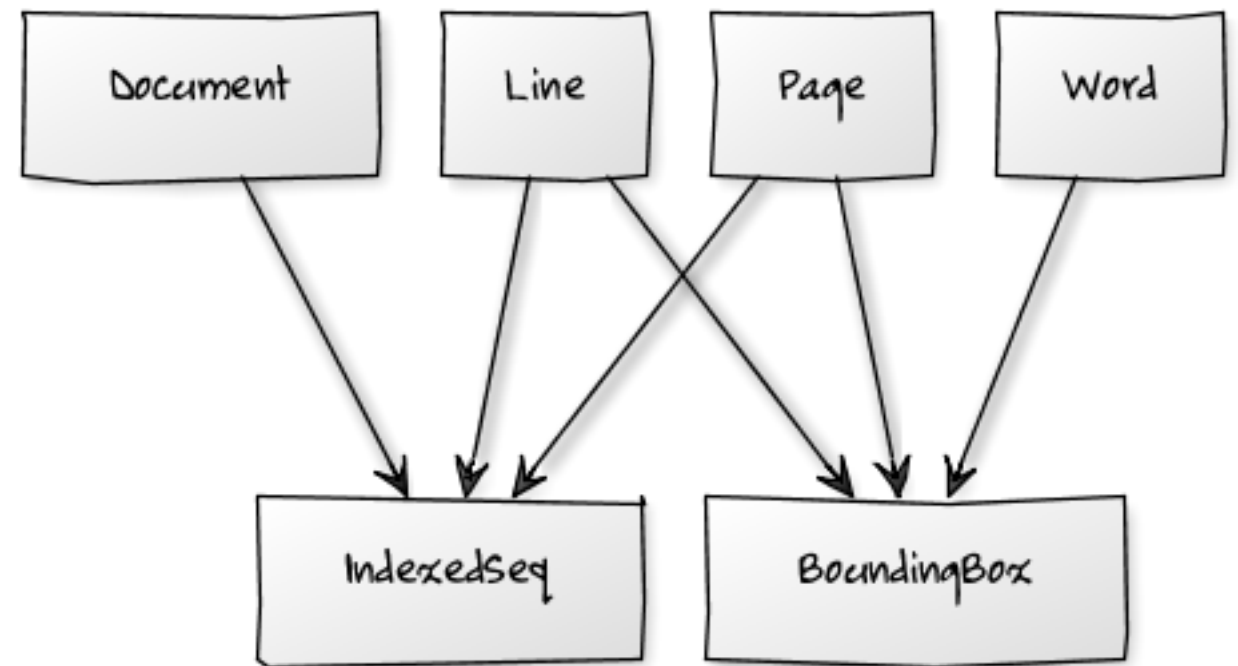
class BoundingBoxSpec extends Specification{
  ...
  "BoundingBoxes in relations to each other" should {
    "give leftDistanceTo" in new testBoxes {
      box2.leftDistanceTo(box1).should_==(2)
    }
    "give rightDistanceTo" in new testBoxes {
      box1.rightDistanceTo(box2).should_==(2)
    }
    "give topDistanceTo" in new testBoxes {
      box2.topDistanceTo(box1).should_==(3)
    }
    "give bottomDistanceTo" in new testBoxes {
      box1.bottomDistanceTo(box2).should_==(3)
    }

    "Test weather a box encloses another Box" in new testBoxes {
      bigBox encloses box1 should beTrue
    }
    "Test weather a box is encloled by another box" in new testBoxes {
      box2 enclosed_by bigBox should beTrue
    }
    "Enclosing is idempotent" in new testBoxes {
      bigBox enclosed_by bigBox should beTrue
    }
  }
  ...
  trait testBoxes extends Scope{
    val box1 = new BoundingBox { val coordinates = ((2,1),(10,5)) }
    val box2 = new BoundingBox { val coordinates = ((12,8),(18,14)) }
    val bigBox = new BoundingBox { val coordinates = ((0,0),(100,50)) }
  }
}

```

# Kurzzusammenfassung

- Durch die skizzierte Implementierung kann das Dokument gemäß seinem physischen Aufbau durchlaufen werden
- Nach dem (nicht mehr UML-konformen Diagramm) sind nun Zeilen, Wörter und Seiten auch Bounding Boxes die in Relationen zueinander gesetzt werden können
- Beispielanfrage: Ist Wort 4 aus Zeile 5 rechts von Wort 9 aus Zeile 3



```
val page =  
Document.fromFolder("example").pages(5) //  
Beispielseite  
  
val w1 = page.lines(4).words(4)  
val w2 = page.lines(2).words(8)  
  
w1 rightOf( w2 )
```

# Globale Betrachtungen zu fundamentalen Layouteinheiten unter Nutzung des Dokumentobjekts

# Fundamentale Layouteinheiten und deren Aufgaben

---

- Annahme 1: In einem OCR-Prozess gewonnene Daten zeichnen sich durch eine gewisse Ungenauigkeit aus
- Annahme 2: Man kann damit rechnen, dass trotz gewisser Ungenauigkeit, bei einer ausreichend großen Datenbasis, sich sogenannte Spitzen ergeben, die dem „exakten“ Wert entsprechen und abweichende Werte mit mehr Abstand zu den Spitzen weniger häufig auftreten. Dies führt zur Bildung sogenannter „Berge“.
- Annahme 3: Layout ist nicht zufällig und trägt semantische Information. Somit ist ein Berg gleichbedeutend mit einer fundamentalen Layouteinheit.
- Annahme 4: Die Bedeutung/Aufgabe einer fundamentalen Layouteinheit kann von Werk zu Werk variieren.

# Ein Beispiel: fundamentale Layouteinheiten durch Abstand der Elemente zum linken Rand

---

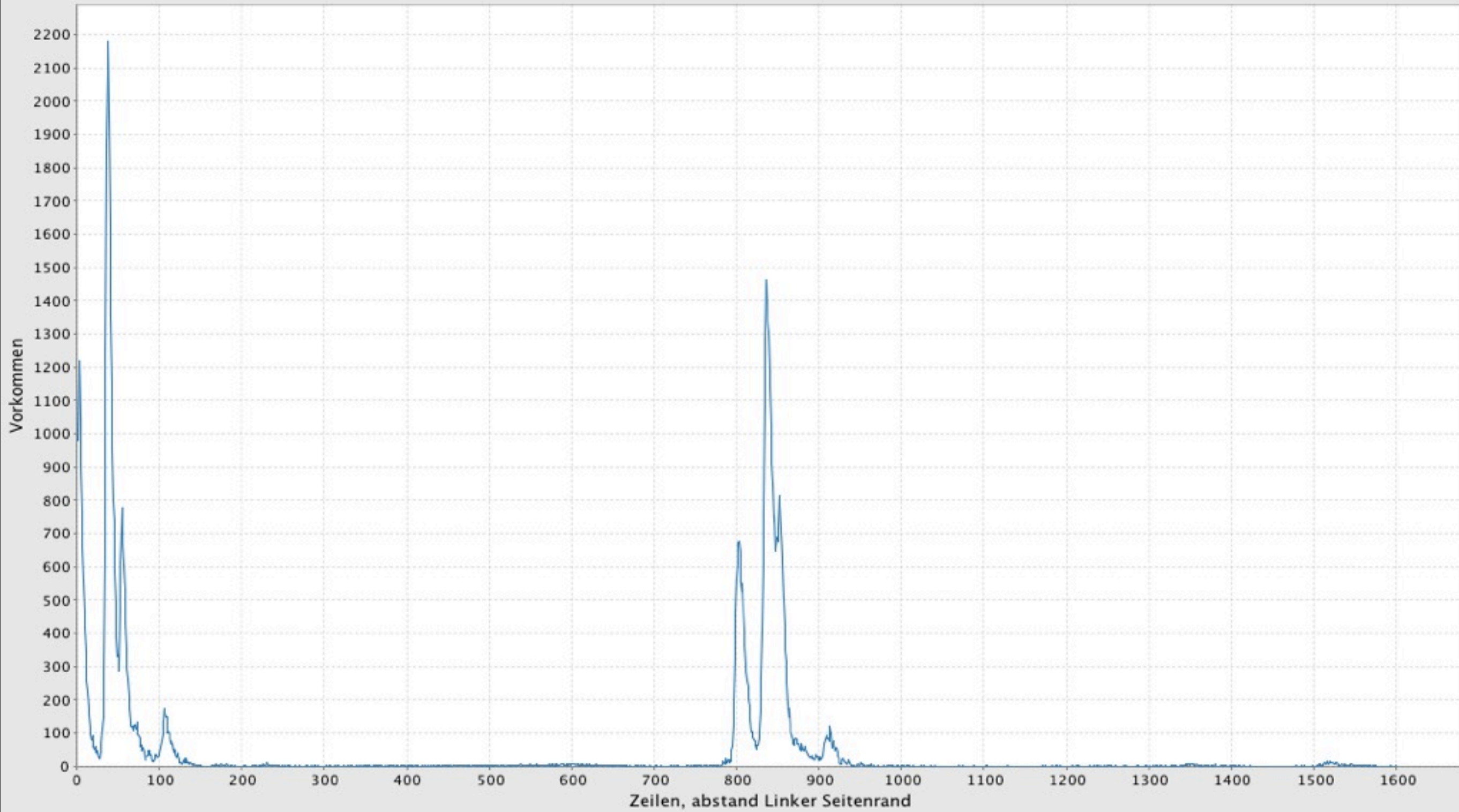
- Folgende Kurven wurde gewonnen indem man den Abstand jeder einzelnen Zeile zum linken Seitenrand auf der x-Achse und die Häufigkeit des Auftretens auf der y-Achse verzeichnet
- Mithilfe der vorher skizzierten API kann man alle linken Seitenabstände relativ intuitiv abfragen:

```
// Dokument einlesen
val doc = Document.fromFolder("testdoc")

// Frequenz aller Linken Seitenabstände erstellen
doc.lines groupBy(x => x) map{ case(left,elements) =>
(left,elements.size) }
```

- Transformiert man das Ergebnis in eine grafische Darstellung erhält man folgendes Bild:

## Zeilenränder





**Alèzes, Alaises** *f. pl.* d'un fourreau de sabre en métal (fût en bois). *Die Holzwände.* Lining.

**Algèbre** *f.* (Math.) *Die Algebra, Buchstabenrechnung.* Algebra, literal calculus.

**Alichon** *m.* d'une roue de moulin à eau. *Die Schaufel.* Float-board. *Compar.* Aube d'une roue.

**Alidade** *f.* (règle mobile autour d'un cercle gradué pour mesurer des angles (Géom.) *Das Diopterlineal, die Alhidade.* Alhidada, Alhidade.

**Aligner** *v. a.* (dresser ou ranger sur une ligne droite, tirer une ligne droite). *Eine gerade Linie abstecken, abschnüren.* To line out, to mark out.

**Aligner le bois** (marquer la charpente au cordeau), **Tringler** la charpente. *Das Holz schnüren oder abschnüren.* To line out stuff, to strike a line.

**Aligner v. a. la pièce, Diriger v. a. la ligne de mire** (Artill.) *Die Linie nehmen, dem Geschütz die Seitenrichtung geben.* To give the direction to a gun, to take the line of direction.

**Aligner v. a. un terrain, Jalonner v. n.** (le marquer avec des piquets, pour le mesurer à la chaîne etc.) (Géod.) *Gerade Linien abstecken.* To mark out lines.

**Alimentaire** *adj.* (tout ce qui appartient à l'alimentation de la chaudière) (Mach.) *Zur Speisung gehörig.* What belongs to feeding (the boiler). *Voy.* Pompe alimentaire.

**Alimentation** *f.* de la chaudière (action d'alimenter) (Mach.) *Die Speisung.* Feeding.

**Alimenter** *v. a.* une chaudière (remplacer l'eau à mesure qu'elle s'échappe sous forme de vapeur) (Mach.) *Speisen.* To feed.

**Alinéa m., Section f.** (Impr.) *Der Absatz, die neue Zeile, der Abschnitt.* Paragraph.

**Alisé** *adj.* *Voyez* Alizé.

**Alisier** *m.* (bois de l'arbre *Cratægus aria*). *Das*

**Aller à la bouline, Bouliner** *v. n.* (Mar.)

*Bei (in schiefer Richtung von) dem Winde segeln oder halten, den Kurs bei dem Winde nehmen.* To sail with a scant wind, to close to the wind.

**Aller au lof** (Mar.) *Anluven (das Schiff so drehen, dass der Wind schon etwas von vorn kommt).* To go to windward, to go to the weather side.

**Aller de conserve** (Mar.) *Unter Admiralschaft segeln, Admiralschaft machen.* To sail in company.

**Aller debout au vent** (Mar.) *Flach in den Wind segeln.* To sail right in the wind's eye, to sail head to wind.

**Aller en lest ou sur son lest** (Mar.) *Nur ballastbeladen sein.* To go on the ballast.

**Aller entre deux écoutes** (avoir le vent en poupe) (Mar.) *Zwischen zwei Halsen fahren, mit offenen Halsen segeln.* To sail with both sheets aft, to sail right before the wind.

**Aller vent grand largue** (Mar.) *Raumschoots (mit Backstagswind) segeln, Raumschoots-Wind haben.* To sail with a quartering wind.

**Alliage** *m.* (union de deux ou de plusieurs métaux au moyen de la fusion) (Métal.) *Die Legirung, der Zusatz zum Legiren.* Alloy, alloy.

**Alliage m. d'or et d'iridium.** *Das Iridgold.* Aurate of iridium.

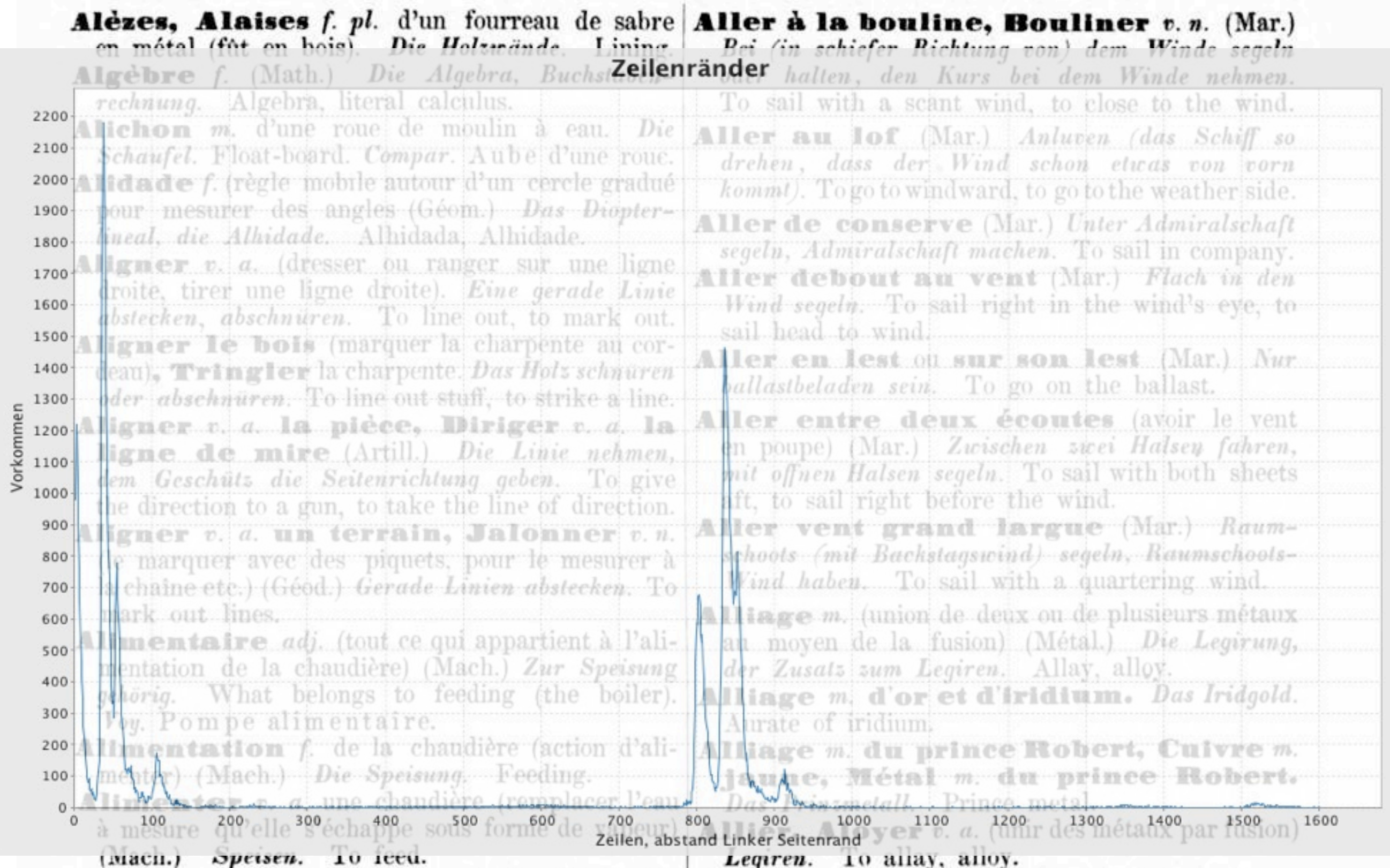
**Alliage m. du prince Robert, Cuivre m. jaune, Métal m. du prince Robert.** *Das Prinzmetall.* Prince metal.

**Allier, Aloyer** *v. a.* (unir des métaux par fusion) *Legiren.* To alloy, alloy.

**Allizari, Allizarine.** *Voyez* Alizari etc.

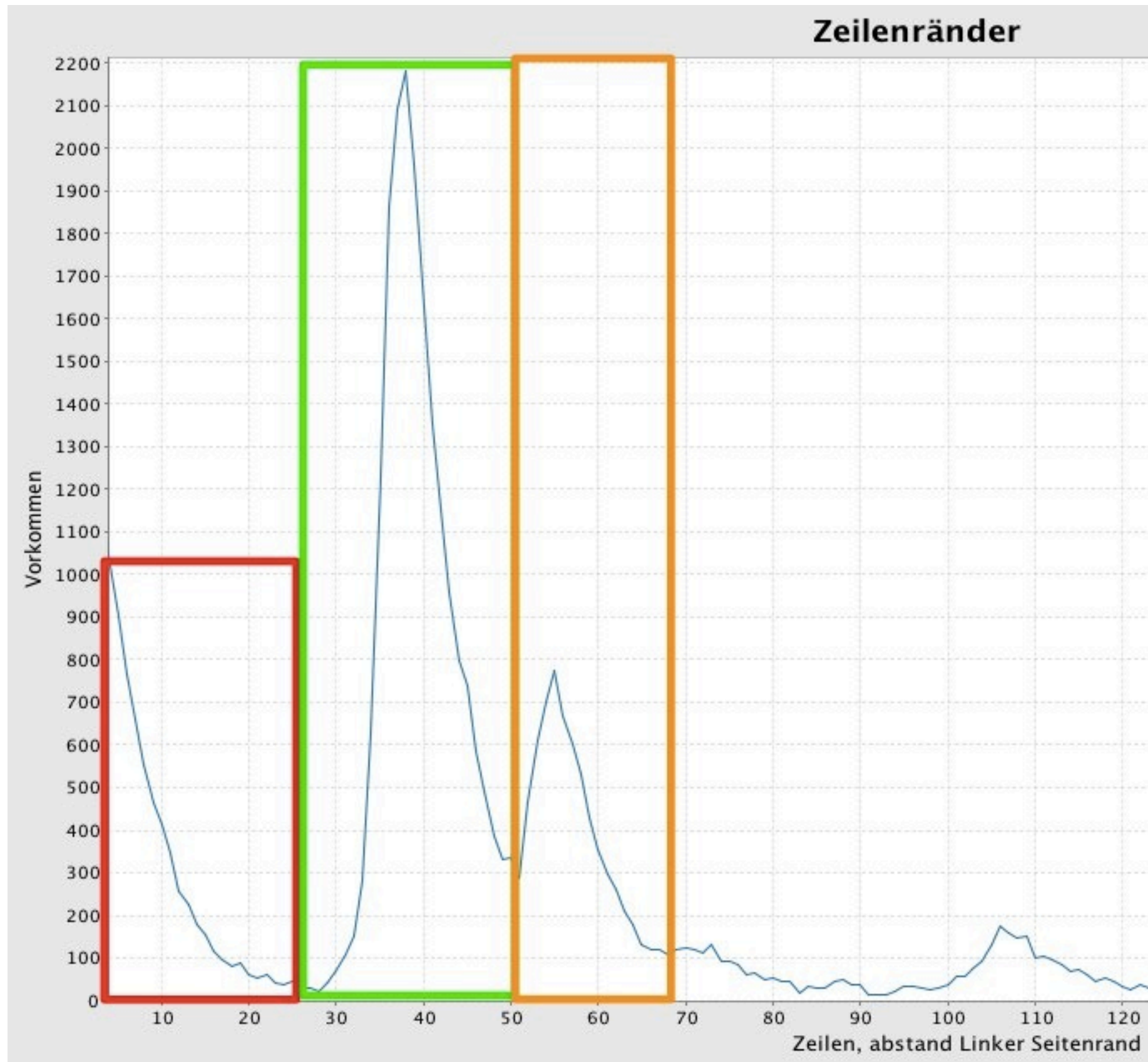
**Allochroïte** *f.* (variété de grenat, d'une couleur grisâtre, jaunâtre ou rougeâtre, qui se trouve en Norwége) (Minér.) *Der Allochroit.* Allochroite.





**Allizari, Allizarine.** *Voyez Alizari etc.*

**Allochroïte** *f.* (variété de grenat, d'une couleur grisâtre, jaunâtre ou rougeâtre, qui se trouve en Norwége) (Minér.) *Der Allochroit. Allochroite.*





**Alèzes, Alaises** *f. pl.* d'un fourreau de sabre en métal (fût en bois). *Die Holzwände.* Lining.

**Algèbre** *f.* (Math.) *Die Algebra, Buchstabenrechnung.* Algebra, literal calculus.

**Alion** *m.* d'une roue de moulin à eau. *Die Schaufel.* Float-board. *Compar.* Aube d'une roue.

**Alidade** *f.* (règle mobile autour d'un cercle gradué pour mesurer des angles (Géom.) *Das Dioptrical, die Alhidade.* Alhidada, Alhidade.

**Aligner** *v. a.* (dresser ou ranger sur une ligne droite, tirer une ligne droite). *Eine gerade Linie abstecken, abschnüren.* To line out, to mark out.

**Aligner le bois** (marquer la charpente au cordeau), **Tringler** la charpente. *Das Holz schnüren oder abschnüren.* To line out stuff, to strike a line.

**Aligner** *v. a.* **la pièce, Diriger** *v. a.* **la ligne de mire** (Artill.) *Die Linie nehmen, dem Geschütz die Seitenrichtung geben.* To give the direction to a gun, to take the line of direction.

**Aligner** *v. a.* **un terrain, Jalonner** *v. n.* (le marquer avec des piquets, pour le mesurer à la chaîne etc.) (Géod.) *Gerade Linien abstecken.* To mark out lines.

**Alimentaire** *adj.* (tout ce qui appartient à l'alimentation de la chaudière) (Mach.) *Zur Speisung gehörig.* What belongs to feeding (the boiler). *Voy. Pompe alimentaire.*

**Alimentation** *f.* de la chaudière (action d'alimenter) (Mach.) *Die Speisung.* Feeding.

**Alimenter** *v. a.* une chaudière (remplacer l'eau à mesure qu'elle s'échappe sous forme de vapeur) (Mach.) *Speisen.* To feed.

**Alinéa** *m.*, **Section** *f.* (Impr.) *Der Absatz, die neue Zeile, der Abschnitt.* Paragraph.

**Alisé** *adj.* *Voyez Alizé.*

**Alisier** *m.* (bois de l'arbre *Crataegus aria*). *Das Mehlbeerbaumholz.* White hawthorn.

**Alisier, Alouchier** *m.* (bois de l'arbre *Crataegus*

**Aller à la bouline, Bouliner** *v. n.* (Mar.) *Bei (in schiefer Richtung von) dem Winde segeln oder halten, den Kurs bei dem Winde nehmen.* To sail with a scant wind, to close to the wind.

**Aller au lof** (Mar.) *Anluven (das Schiff so drehen, dass der Wind schon etwas von vorn kommt).* To go to windward, to go to the weather side.

**Aller de conserve** (Mar.) *Unter Admiralschaft segeln, Admiralschaft machen.* To sail in company.

**Aller debout au vent** (Mar.) *Flach in den Wind segeln.* To sail right in the wind's eye, to sail head to wind.

**Aller en lest** ou **sur son lest** (Mar.) *Nur ballastbeladen sein.* To go on the ballast.

**Aller entre deux écoutes** (avoir le vent en poupe) (Mar.) *Zwischen zwei Halsen fahren, mit offenen Halsen segeln.* To sail with both sheets aft, to sail right before the wind.

**Aller vent grand largue** (Mar.) *Raumschoots (mit Backstagswind) segeln, Raumschoots-Wind haben.* To sail with a quartering wind.

**Alliage** *m.* (union de deux ou de plusieurs métaux au moyen de la fusion) (Métal.) *Die Legirung, der Zusatz zum Legiren.* Alloy, alloy.

**Alliage** *m.* **d'or et d'iridium.** *Das Iridgold.* Aurate of iridium.

**Alliage** *m.* **du prince Robert, Cuivre** *m.* **jaune, Métal** *m.* **du prince Robert.** *Das Prinzmetall.* Prince metal.

**Allier, Aloyer** *v. a.* (unir des métaux par fusion) *Legiren.* To alloy, alloy.

**Allizari, Allizarine.** *Voyez Alizari etc.*

**Allochroïte** *f.* (variété de grenat, d'une couleur grisâtre, jaunâtre ou rougeâtre, qui se trouve en Norwége) (Minér.) *Der Allochroit.* Allochroite.

**Allonge** *f.* *Voyez Alonge.*

**Allonger** *v. n.* (d'engranger du bois fixé sur



# Beispiel aus hOCR (in Entwicklung)

	Adoniram. — Agariter.	11
Adoniram.	Der Herr ist hoch. Ein Sohn Abda, und der Rentmeister zu Salomos Zeiten 1 Kön. 4, 6; 5, 14., heißt auch Adoram, d. i. großmächtig, 1 Kön. 12, 18. und Hadoram 2 Chron. 10, 18.	
Adoni Zedek.	Der Herr zu Zedek. War König zu Jerusalem. Jer. 10, 1.	
Ador oder Dor,	d. i. Herberge, Wohnung. 1 Mac. 13, 20.	
Adreaim.	Doppelte Herrlichkeit, doppelte Pracht. Eine Stadt, welche Rehabeam gebauet hat. 2 Chron. 11, 9.	
Adramelech.	Des Königs ist die Herrlichkeit. 1) Ein Göze des Sephariten. 2 Kön. 17, 31. 2) Ein Sohn Sanheribs, des Königs in Assyrien. 2 Kön. 19, 37; Jes. 37, 38.	
Adramitus od. Adramita,	eine Seestadt in Afrika, davon Lucas ein adramitisches Schiffes erwähnt, auf welchem Paulus Schiffbruch litt. Ap. Gesch. 27, 2.	

# Interpretation

---

- Die Berge korrespondieren mit den Zeilenanfänge
- Das zweispaltige Layout spiegelt sich in der Bergstruktur wieder
- Es existieren drei Level von Einrückungen
- Das seltenste Level korrespondiert mit den Lemmazeilen
- Die tiefste Einrückung kennzeichnet Sub-Einträge

# Weitere Anwendungen

---

- Rekonstruktion von bold-Character, die das OCR nicht erkannt hat
- ...