1. Exercise

TECHNISCHE UNIVERSITÄT DARMSTADT

The first exercise

Knowlege Engineering
Fachbereich Informatik
Technische Universität Darmstadt

Exercise Presentation:
Frank Englert
Jens Haase

2. Exercise

Overview

1. Language Detection via character distribution
   - How it works
   - Results of the language detection challenge
2. Web crawler
   - Introduction
   - New URLs found
   - URLs per Page Statistics
   - Frequency of links
   - Further results
   - Experiences

Task 1 - Language detection

Language Detection via letter distribution

- The Firefox Plugin uses two detection modes
  - Via letter frequency analysis
  - Via syllable frequency analysis
- The language detection algorithm is the same for both cases
- Advantages of using two detection modes:
  - Double check the language detection results
  - Collect information which mode works better
- The Source of the frequency tables is http://bit.ly/jZHf0H

Task 1 - Language detection

Letter frequency revisited

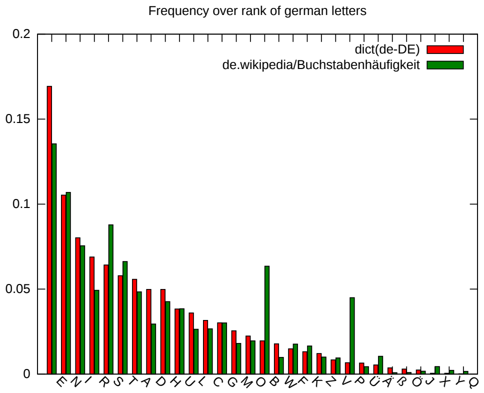


Abbildung: The letter frequency of the dict compared to the frequency of an exemplary web site

Task 1 - Language detection

Syllable frequency revisited

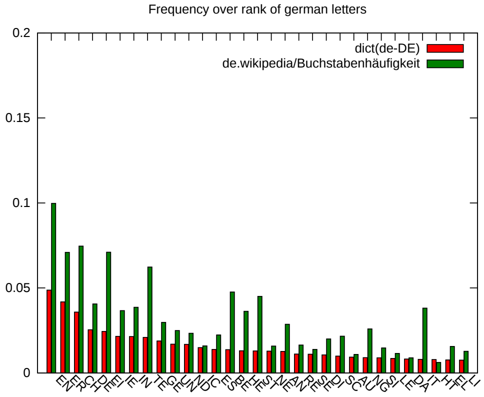


Abbildung: 2. The syllable frequency of the dict compared to the frequency of an exemplary web site

Task 1 - Language detection

Ranking Results of letters and syllables

rank	1	2	3	4	5	6	7	8	9	10
Dict	E	N	I	R	S	T	A	H	D	U
Example	E	N	S	I	T	B	R	A	P	H

Tabelle: Letter Ranking Results of de.wikipedia.org/Buchstabenhäufikeit vs the german average

rank	1	2	3	4	5	6	7	8	9	10
Dict	EN	ER	CH	DE	EI	IE	IN	TE	GE	UN
Example	EN	CH	EI	ER	TE	BE	ST	DE	IN	IT

Tabelle: Syllable Ranking Results of de.wikipedia.org/Buchstabenhäufikeit vs the german average

Task 1 - Language detection

Results of the frequency analysis

- Fazit of a first analysis:
  - Don’t use the letter or syllable probabiliy itself
    - It might work for letters as you can see in picture 1
    - But the variance for the syllables is to high
    - So it will fail for syllables like in picture 2
  - Only use the rank. It matches better as you can see in the slide before.
  - But weight it. The letters with the highest probability in the dict should have the highest impact on the rank.
- **Calculate the sum of the weighted rank for each language**
- **Then take the language with lowest weight as estimation**

Task 1 - Language detection

Algorithm details

- **The algorithms works with the following steps**
- A chunk is either a letter or a syllable
- dict contains the most important chunks of a language sorted by rank
  1. Take the text an split it to chunks(letters or syllables)
  2. Remove all chunks which are not in the dict
  3. Count the chunks and sort them by the count value. The result of this step is further called rankedChunks
  4. The weighted difference between the dictionary and the rankedChunks is
     - $\sum_{i=0}^{len(dic)} \frac{|i - rankedChunks.indexOf(dict[i])|}{log_2(i+2)}$
     - If dict and rankedChunks are equals the weighted difference is 0
- repeat the steps 1-4 for all available languages. Take the language with the lowest rank.

Task 1 - Language detection

Results of the language challenge

Rank	letter lang	syllable lang
1	englisch	-
2	englisch	-
3	deutsch	-
4	französisch	-
5	deutsch	-
6	deutsch	deutsch
7	französisch	französisch
8	französisch	französisch
9	englisch	englisch
10	deutsch	deutsch

Tabelle: Detection results of the firefox plugin

Task 1 - Language detection

Further improvement

- **Easy:**
  - Add more languages
- **A lot of work:**
  - The Plugin checks already p, div and span tags. But Ajax Pages still doesn’t work well.
  - Try to estimate the best detection result if the syllable and the letter mode returns different results
- **Most Interesting:**
  - Improve the weighting algorithm to reduce the amount of needed text
  - Implement a learning mode to train new languages

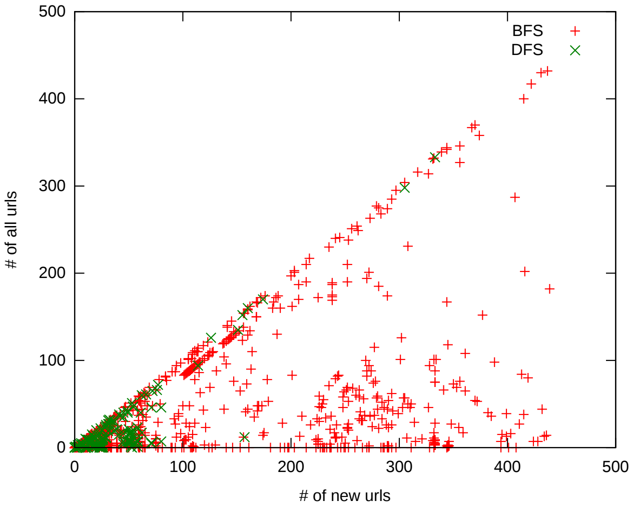
Task 2 - Crawler

Introduction

- **Implemented in Scala**
- **Currently, runs in a single thread.**
  - Therefore we need not to worry about to many access on the same host
  - But it can be easily moved to multithread with the Akka middleware
- **Started crawling at** `http://news.google.de`
- **Indexed 1000 pages with BFS queue and DFS queue**

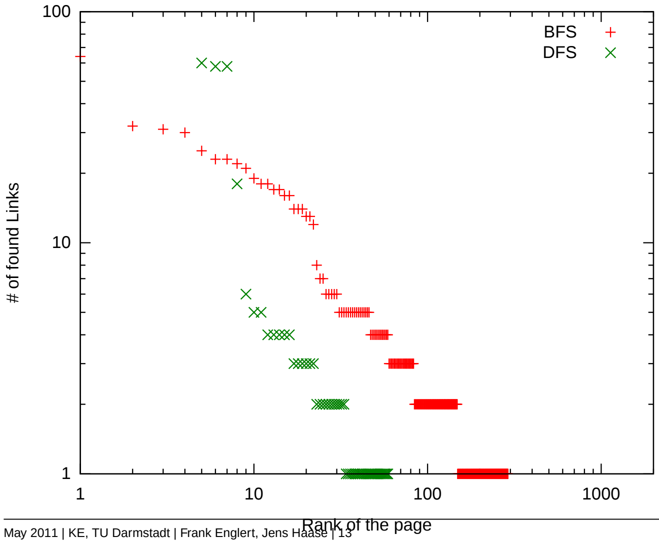
Task 2 - Crawler

New URLs found



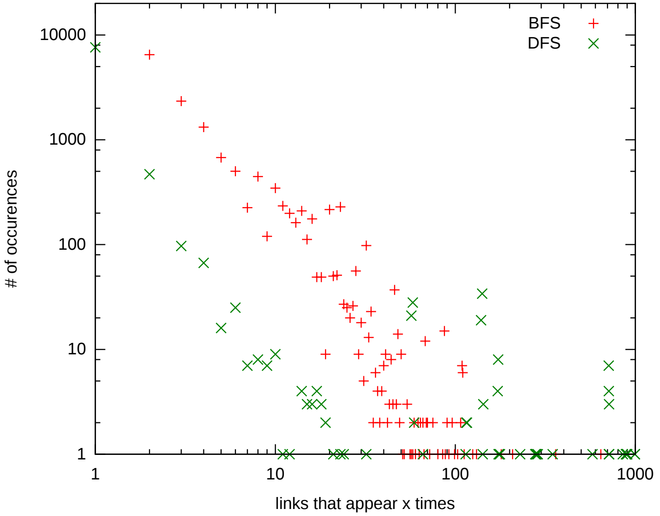
Task 2 - Crawler

URLs per Page Statistics



Task 2 - Crawler

Frequency of links



Task 2 - Crawler

Further results

BFS	DFS
136	20

Tabelle: Different Hosts found

Language	DFS	BFS
german	55	*935*
english	*943*	43
french	0	1
portuguese	0	1
<unknown>	2	20
$\sum$	1000	1000

Tabelle: Found languages in all pages

Task 2 - Crawler

Experiences

- **With Javas `URL` class a URL can easily brought to the same form**
- **But it has problems with `javascript:` and other *”protocols”***
- **Solution: simple wrap with a try catch block**
- **For crawling exception handling is a must! Else the crawler will stop in near time**