

# BlackLab: a researcher's best friend

Jan Niestadt, INL

github.com/INL/BlackLab @BlacklabINL





# retrieval engine



retrieval engine (woof!)



retrieval engine for annotated text.



multi-input retrieval engine for annotated text.



# multi-input, multi-query-language, retrieval engine for annotated text.



easy, multi-input, multi-querylanguage, retrieval engine for annotated text.



easy, multi-input, multi-querylanguage, complex retrieval engine for annotated text.



easy, multi-input, multi-querylanguage, complex retrieval and analysis engine for annotated text.



fast, scalable, easy, multi-input, multiquery-language, complex retrieval and analysis engine for annotated text.



open source, fast, scalable, easy, multi-input, multi-query-language, complex retrieval and analysis engine for annotated text.



open source, fast, scalable, easy, multi-input, multi-query-language, complex retrieval and analysis engine for annotated text written in Java.



open source, fast, scalable, easy, multi-input, multi-query-language, complex retrieval and analysis engine for annotated text written in Java using Apache Lucene.



Actively developed, open source, fast, scalable, easy, multi-input, multi-query-language, complex retrieval and analysis engine for annotated text written in Java using Apache Lucene.



Actively developed, open source, fast, scalable, easy, multi-input, multi-query-language, complex retrieval and analysis engine for annotated text written in Java using Apache Lucene with a bright future.



Active y developed, open source, fast, scales of multi-input, multi-in



## verb like a adj noun

"His heart was spinning like a broken compass."



TEI
Any XML

Any text

PageXML

FoLiA



Sketch SRU/CQL Your own Lucene CWB CQP QL Google-like



# Simple word query:

"labrador"



# Regular expression:

".+rad.\*"



#### Sequences of words:

"quick" "black" "labrador"



#### Word properties:

[pos = "adj"] [word != "yellow"] "labrador"



#### Token-level regex operators:

```
[pos = "adj"]+ "labrador"
"quick" "black"? "labrador"
"b.+"{2,} "labrador"
```



#### Matchall operator:

```
"quick" []{1,3} "labrador" []{3}
```

```
// distance
// tri-grams
```



#### XML tag search:

```
<ne> "Lassie" </ne>
"Fido" within <ne/>
<ne/> containing "Rover"
```



#### More on the way:

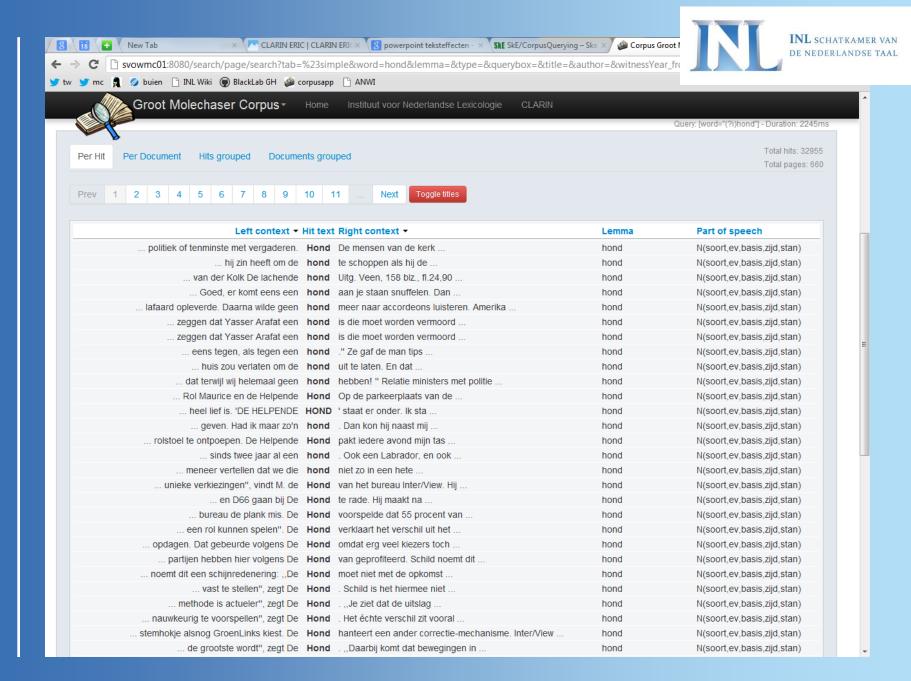
1:[] "dog" "and" 2:[] "cat" & 1.word = 2.word

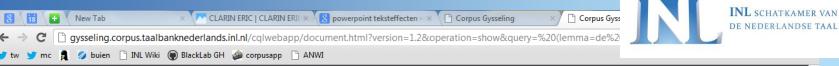
(matches "black dog and black cat")

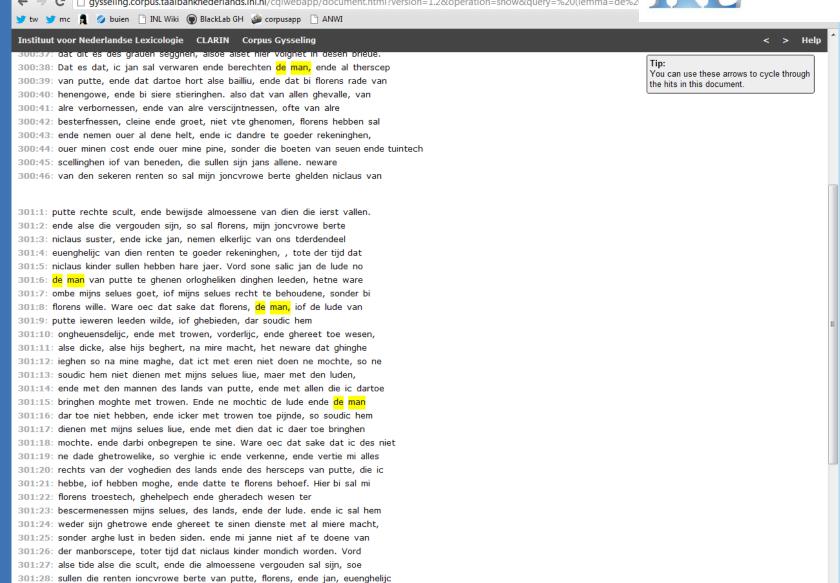


# Want more ways to query?

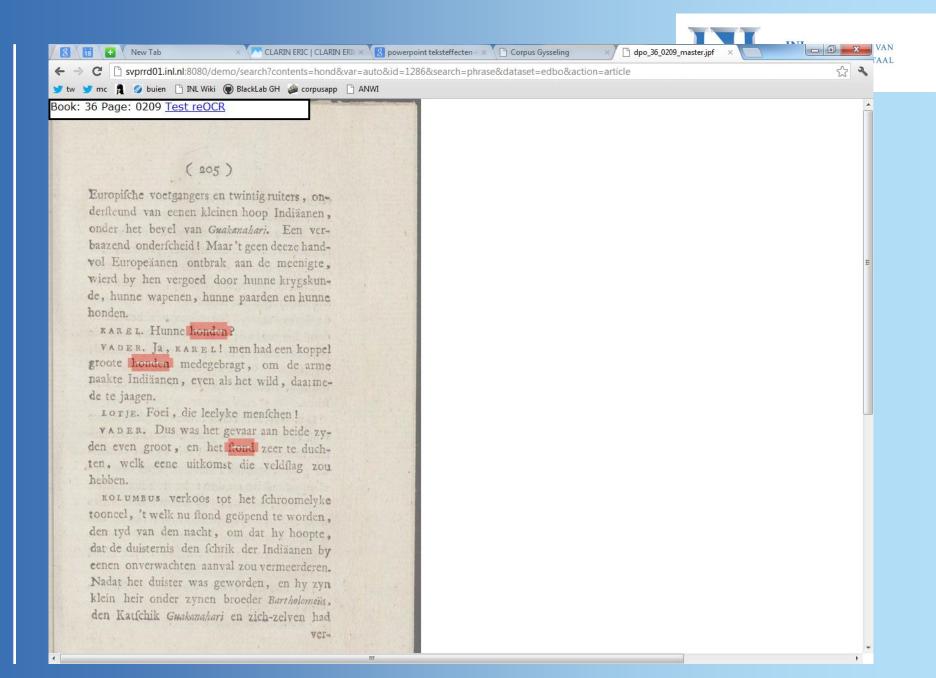
Let us know!



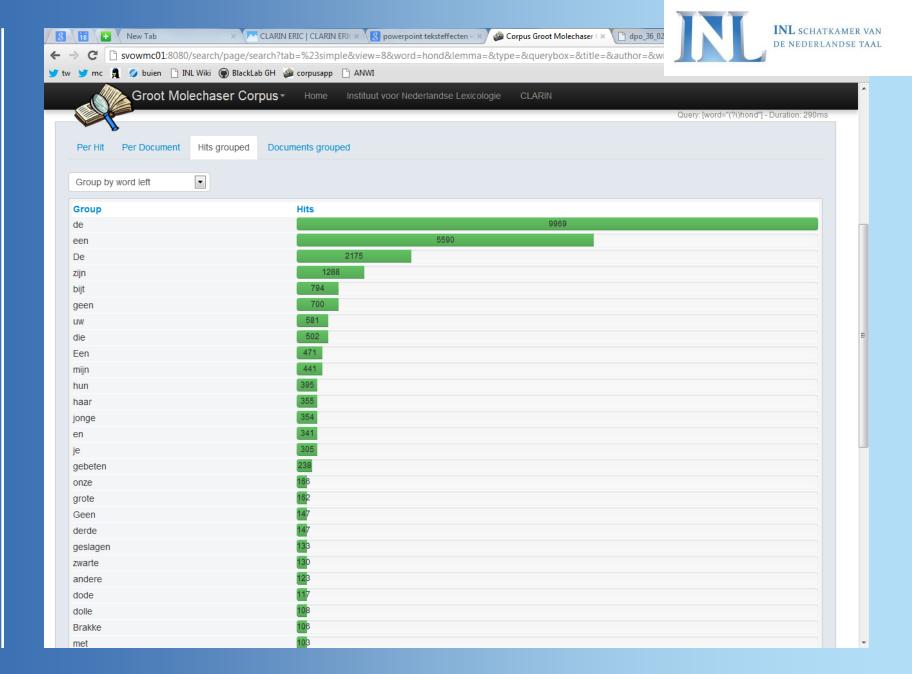




301:29: delen harelijc tderdendeel, toter tijd dat niclaus kinder mondich









#### Works in progress:

- Random sampling
- Collocations
- More cool tools planned!



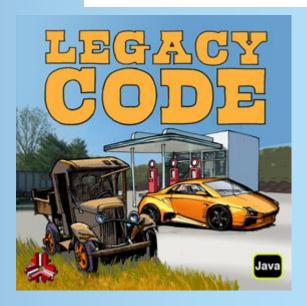






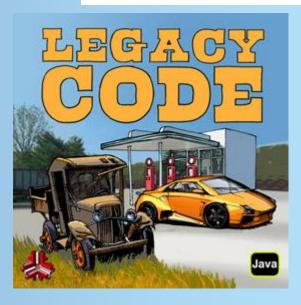




















Delivery 1 Delivery 2 Delivery 3







Incremental plan





Delivery 2

Delivery 3

















Search / operation	# Results	Search time
"stad"	± 190,000	< 1 S
"de" "stad"	± 120,000	2 S
"de" "sta.*"	± 370,000	6 s
"de" "grote"? "stad"	± 120,000	5 S
"die.*"{2,}	± 29,000	10 S
"water" / group by word left	± 140,000	< 1 S
"boven" / sort by left context	± 160,000	< 1 S
"onder" / group by word right	± 780,000	6 s



## Scalability / speed:

- Lucene: ++
- Sorting/grouping: ++
- Displaying results: +
- Distributed: +/- (future)





Forward Index Content Store

Finding hits (SpanQuery classes, many custom)

Fast sorting / grouping on context; collocations

Displaying results



Lucene Index

Finding hits (SpanQuery classes, many custom)

Forward Index

Fast sorting / grouping on context; collocations

Content Store

Displaying results

Idea: integrate these two to improve speed and reduce disk/memory usage



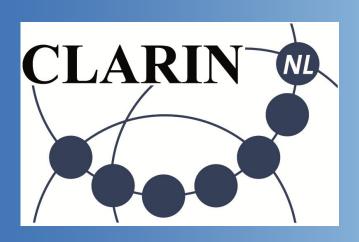


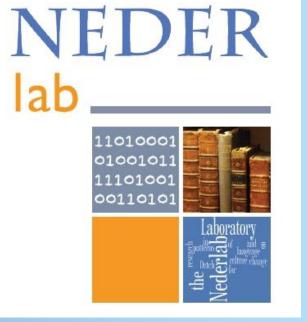
Improving Access to Text

## **IMPACT**















1



Data in supported format (ask us!)





Data in supported format (ask us!)







Data in supported format (ask us!)

BlackLab index; Test with QueryTool

```
public class BlackLabDemoServlet extends HttpServlet {
    Searcher searcher;
    @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        try {
            // Parse and execute guery
            String guery = reguest.getParameter("guery");
            TextPattern pattern = CorpusQueryLanguageParser.parse(query);
            Hits hits = searcher.find(pattern);
           // Output hits with a little context
            ServletOutputStream out = response.getOutputStream();
            for (Hit hit: hits) {
               Concordance conc = hits.getConcordance(hit);
                out.println(conc.left + "<b>" + conc.hit + "</b>" + conc.right + "<br/>");
        } catch (Exception e) {
            throw new ServletException(e);
    @Override
   public void init(ServletConfig config) throws ServletException {
            searcher = new Searcher(new File("/home/jan/testindex"));
        } catch (Exception e) {
            throw new ServletException(e);
    @Override
   public void destroy() {
        searcher.close();
```



## The Future NEXT EXIT





## Thank you! Questions?

Jan.Niestadt@inl.nl

Twitter: @BlackLabINL

GitHub: /INL/BlackLab