Die Hard 1.1024.0: Backward compatibility of a search engine with persistent IDs

deRSE19 - Conference for Research Software Engineers in Germany, 2019-06-04

Thomas Krause (Humboldt-Universität zu Berlin)

Stephan Druskat (Friedrich Schiller University Jena, German Aerospace Center (DLR))

Background

The Hexatomic project

“A minimal infrastructure for the **sustainable** provision of extensible *multi-layer annotation software for linguistic corpora*”

- Funded under the call “Research Software Sustainability” issued by DFG under grant number GA 1288/11-1
- Runs from October 2018 until September 2021

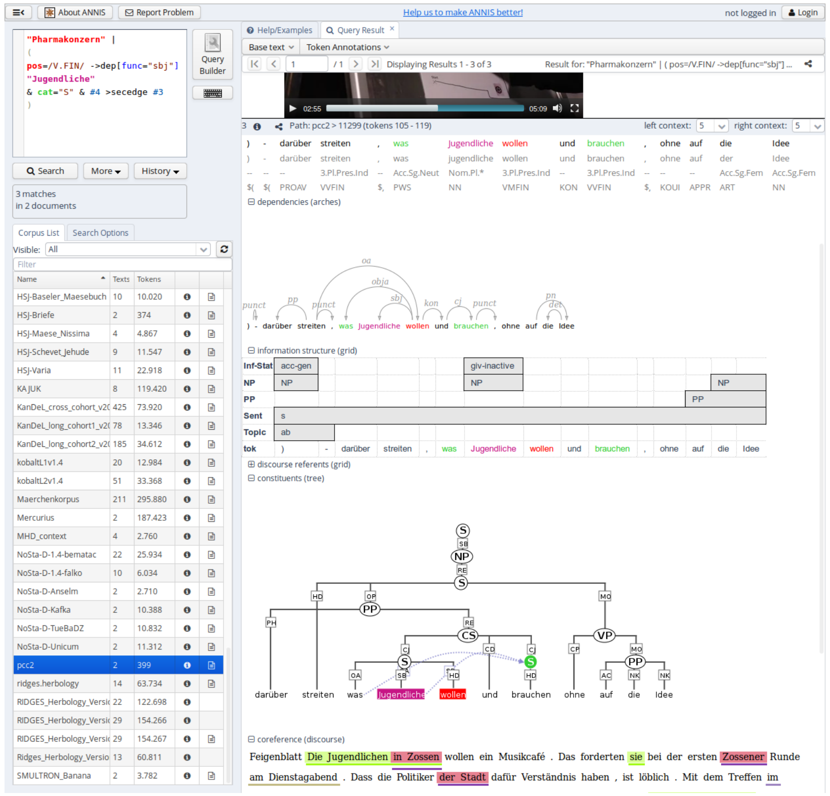
The Hexatomic project

“A minimal infrastructure for the **sustainable** provision of extensible *multi-layer annotation software for linguistic corpora*”

- Funded under the call “Research Software Sustainability” issued by DFG under grant number GA 1288/11-1
- Runs from October 2018 until September 2021
- Thomas Krause: computer scientist who slipped into linguistics
- Stephan Druskat: English M.A. turned software developer & computer scientist
- Both: Research Software Engineers

ANNIS and its query language

Web browser-based search and visualization architecture for *linguistic corpora* with diverse types of *annotation*. Part of the corpus-tools.org collection of tools for linguists. (Druskat et al. 2016)



```
"Pharmakonzern" |
(
pos=/V.FIN/ ->dep[func="sbj"]
"Jugendliche"
& cat="S" & #4 >secedge #3
)
```

3 matches
in 2 documents

Visible: All

Name	Texts	Tokens		
HSJ-Baseler_Maesebuch	10	10.020	●	🗎
HSJ-Briefe	2	374	●	🗎
HSJ-Maese_Nissima	4	4.867	●	🗎
HSJ-Schevet_Jehude	9	11.547	●	🗎
HSJ-Varia	11	22.918	●	🗎
KA JUK	8	119.420	●	🗎
KanDeL_cross_cohort_v20	425	73.920	●	🗎
KanDeL_long_cohort1_v20	78	13.346	●	🗎
KanDeL_long_cohort2_v20	185	34.612	●	🗎
kobaltL1v1.4	20	12.984	●	🗎
kobaltL2v1.4	51	33.368	●	🗎
Maerchenkorpus	211	295.880	●	🗎
Mercurius	2	187.423	●	🗎
MHD_context	4	2.760	●	🗎
NoSta-D-1.4-bematac	22	25.934	●	🗎
NoSta-D-1.4-falko	10	6.034	●	🗎
NoSta-D-Anselm	2	2.710	●	🗎
NoSta-D-Kafka	2	10.388	●	🗎
NoSta-D-TueBaDZ	2	10.832	●	🗎
NoSta-D-Unicum	2	11.312	●	🗎
pcc2	2	399	●	🗎
ridges.herbology	14	63.734	●	🗎
RIDGES_Herbology_Versi	22	122.698	●	
RIDGES_Herbology_Versi	29	154.266	●	
RIDGES_Herbology_Versi	29	154.267	●	🗎
Ridges_Herbology_Versio	13	60.811	●	
SMULTRON_Banana	2	3.782	●	🗎

Displaying Results 1 - 3 of 3 Result for: "Pharmakonzern" | (pos=/V.FIN/ ->dep[func="sbj"] ...

3 Path: pcc2 > 11299 (tokens 105 - 119) left context: 5 right context: 5

)	-	darüber	streiten	,	was	Jugendliche	wollen	und	brauchen	,	ohne	auf	die	Idee
)	-	darüber	streiten	,	was	jugendliche	wollen	und	brauchen	,	ohne	auf	der	Idee
--	--	--	3.Pl.Pres.Ind	--	Acc.Sg.Neut	Nom.Pl.*	3.Pl.Pres.Ind	--	3.Pl.Pres.Ind	--	--	--	Acc.Sg.Fem	Acc.Sg.Fem
$(	$(	PROAV	VVFIN	$,	PWS	NN	VMFIN	KON	VVFIN	$,	KOUI	APPR	ART	NN

Inf-Stat	acc-gen					giv-inactive									
NP	NP					NP							NP		
PP												PP			
Sent	s														
Topic	ab														
tok	)	-	darüber	streiten	,	was	Jugendliche	wollen	und	brauchen	,	ohne	auf	die	Idee

Feigenblatt Die Jugendlichen in Zossen wollen ein Musikcafé . Das forderten sie bei der ersten Zossener Runde am Dienstagabend . Dass die Politiker der Stadt dafür Verständnis haben , ist löblich . Mit dem Treffen im

- Annotations are structured information added to text represented as a **graph with labels**
- Used by expert users (linguists) to **find and analyze linguistic phenomena**
- ANNIS allows finding annotations and *combinations* of annotations with its domain specific query language AQL
- AQL describes **nodes labels** and **joins them with operators,** which constrain the relation of the nodes in the graph

Semantic Versioning

- Popularized by semver.org (Preston-Werner n.d.)
- Explicit statement about compatibility between versions of API
- *MAJOR.MINOR.PATCH*
    - Only bug fixes when *PATCH* changes, API does not change
    - Additions to API marked as increase of *MINOR*
    - Removal and non-backward compatible changes need an increase in *MAJOR*

Semantic Versioning

- Popularized by semver.org (Preston-Werner n.d.)
- Explicit statement about compatibility between versions of API
- *MAJOR.MINOR.PATCH*
  - Only bug fixes when *PATCH* changes, API does not change
  - Additions to API marked as increase of *MINOR*
  - Removal and non-backward compatible changes need an increase in *MAJOR*

Some open questions:

- What is part of the API in a complex piece of software with multiple components?
  - REST API?
  - Query language?
  - Data exchange format?
  - User Interface?

Semantic Versioning

- Popularized by semver.org (Preston-Werner n.d.)
- Explicit statement about compatibility between versions of API
- *MAJOR.MINOR.PATCH*
    - Only bug fixes when *PATCH* changes, API does not change
    - Additions to API marked as increase of *MINOR*
    - Removal and non-backward compatible changes need an increase in *MAJOR*

Some open questions:

- What is part of the API in a complex piece of software with multiple components?
    - REST API?
    - Query language?
    - Data exchange format?
    - User Interface?
- **Do we want to backward-compatible forever? Is there a “1.0 release anxiety”?**

Persistent identifiers (PIDs)

What do I mean exactly when I refer to the “ANNIS software”?

http://corpus-tools.org/annis? https://github.com/thomaskrause/ANNIS/? https://github.com/korpling/ANNIS/? Version 3.5.1? Version 4?

Persistent identifiers (PIDs)

What do I mean exactly when I refer to the “ANNIS software”?

http://corpus-tools.org/annis? https://github.com/thomaskrause/ANNIS/? https://github.com/korpling/ANNIS/? Version 3.5.1? Version 4?

I can reference a specific software by a Digital object identifier (DOI):

DOI 10.5281/zenodo.1212548

Persistent identifiers (PIDs)

What do I mean exactly when I refer to the “ANNIS software”?

http://corpus-tools.org/annis? https://github.com/thomaskrause/ANNIS/? https://github.com/korpling/ANNIS/? Version 3.5.1? Version 4?

I can reference a specific software by a Digital object identifier (DOI):

DOI 10.5281/zenodo.1212548

- In general: resolving an identifier to a resource (digital or not)
- **Should never change**, i.e., you can print it in a book!
- Several systems exist, e.g. DOI, handle.net, ...

Persistent identifiers (PIDs)

What do I mean exactly when I refer to the "ANNIS software"?

http://corpus-tools.org/annis? https://github.com/thomaskrause/ANNIS/? https://github.com/korpling/ANNIS/? Version 3.5.1? Version 4?

I can reference a specific software by a Digital object identifier (DOI):

DOI 10.5281/zenodo.1212548

- In general: resolving an identifier to a resource (digital or not)
- **Should never change**, i.e., you can print it in a book!
- Several systems exist, e.g. DOI, handle.net, ...

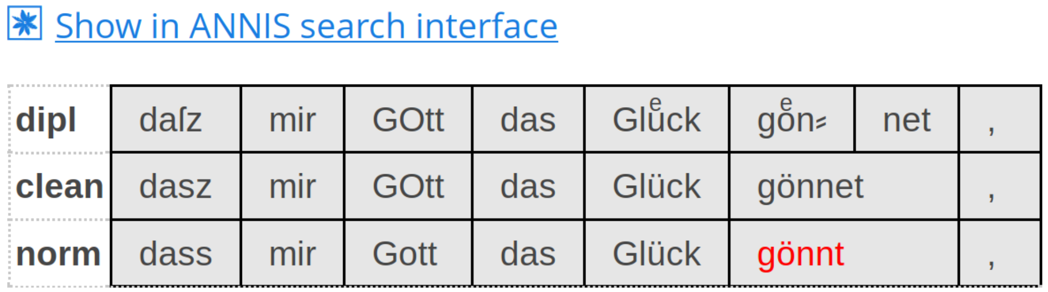
Some open questions:

- If a digital resource moves, who updates the reference?
- Who provides and funds the infrastructure?

Achieving backward compatibility in ANNIS 4

ANNIS reference links

- ANNIS allows generating short links to query results and single matches, e.g., https://korpling.org/annis3/?id=813c3146-2d10-4d0c-8a1f-1b5efc3c051a



Show in ANNIS search interface

dipl	daſz	mir	GOtt	das	Gluͤck	goͤn⸗	net	,
clean	dasz	mir	GOtt	das	Glück	gönnet		,
norm	dass	mir	Gott	das	Glück	gönnt		,

- Glorified URL shortener: expands to a longer URL encoding the match and the actual query paramters, e.g., https://korpling.org/annis3/#_q=bm9ybT0vZ8O2bm50Lw&_c=UklER0VTX[...]
- Query is executed each time the link is opened, no result identifiers are saved

Backward compatibility

Problem:

- ANNIS 3: AQL queries are mapped to SQL queries and executed by PostgreSQL
- ANNIS 4: custom in-memory graph-based search engine written in Rust, which directly executes AQL (Krause 2019)

All old reference links should still work because the query results are part of the research results.

Backward compatibility

Problem:

- ANNIS 3: AQL queries are mapped to SQL queries and executed by PostgreSQL
- ANNIS 4: custom in-memory graph-based search engine written in Rust, which directly executes AQL (Krause 2019)

All old reference links should still work because the query results are part of the research results.

Users literally printed these links in books.

Backward compatibility

Problem:

- ANNIS 3: AQL queries are mapped to SQL queries and executed by PostgreSQL
- ANNIS 4: custom in-memory graph-based search engine written in Rust, which directly executes AQL (Krause 2019)

All old reference links should still work because the query results are part of the research results.

Users literally printed these links in books.

Solution 1: Keep the old software running forever (in parallel to the new one)

Backward compatibility

Problem:

- ANNIS 3: AQL queries are mapped to SQL queries and executed by PostgreSQL
- ANNIS 4: custom in-memory graph-based search engine written in Rust, which directly executes AQL (Krause 2019)

All old reference links should still work because the query results are part of the research results.

Users literally printed these links in books.

Solution 1: Keep the old software running forever (in parallel to the new one)

Solution 2: Make sure that each query that has been referenced, produces the same result in ANNIS 4 as in ANNIS 3

Backward compatibility

Problem:

- ANNIS 3: AQL queries are mapped to SQL queries and executed by PostgreSQL
- ANNIS 4: custom in-memory graph-based search engine written in Rust, which directly executes AQL (Krause 2019)

All old reference links should still work because the query results are part of the research results.

Users literally printed these links in books.

Solution 1: Keep the old software running forever (in parallel to the new one)

Solution 2: Make sure that each query that has been referenced, produces the same result in ANNIS 4 as in ANNIS 3

- Execute each referenced query on both ANNIS 3 and 4
- Compare the results
- If successful: Migrate the links to the new ANNIS 4 installation

Query language incompatibility is a feature

- New version will remove some query language functions and fix bugs in the query execution
- Backward compatibility means supporting old features or even replicating these bugs

Query language incompatibility is a feature

- New version will remove some query language functions and fix bugs in the query execution
- Backward compatibility means supporting old features or even replicating these bugs
- We need a *quirks mode* emulating old behavior

Query language incompatibility is a feature

- New version will remove some query language functions and fix bugs in the query execution
- Backward compatibility means supporting old features or even replicating these bugs
- We need a *quirks mode* emulating old behavior

Other examples:

- Internet Explorer
- Rust language "editions" for opting in into breaking-change features

"We can't get rid of it because we have a commitment to not breaking users' code. There will not be a Rust 2.0." - Steve Klabnik (https://news.ycombinator.com/item?id=19638000)

Selected problems from the migration process

Formalized semantics of the data model and the query language

- In an ideal world a query language is formally defined like Datalog (Ceri, Gottlob, and Tanca 1989)
    - Strictly based on predicate logic
    - Declaration of so-called facts and rules how to infer new facts
- All you need to restore a digital object would be the data and any implementation of the query language conforming to the specification

Formalized semantics of the data model and the query language

- In an ideal world a query language is formally defined like Datalog (Ceri, Gottlob, and Tanca 1989)
    - Strictly based on predicate logic
    - Declaration of so-called facts and rules how to infer new facts
- All you need to restore a digital object would be the data and any implementation of the query language conforming to the specification

Real world: SQL

Formalized semantics of the data model and the query language

- In an ideal world a query language is formally defined like Datalog (Ceri, Gottlob, and Tanca 1989)
    - Strictly based on predicate logic
    - Declaration of so-called facts and rules how to infer new facts
- All you need to restore a digital object would be the data and any implementation of the query language conforming to the specification

Real world: SQL

- SQL versions have been standardized (..., SQL-93, SQL:1999, SQL:2003, ...)
- Various implementations (MySQL, PostgreSQL, Oracle, DB2, SQLite, ...) with different support for the standard and vendor extensions
- AQL has only two implementations, but the first implementation inherited semantics of SQL and its implementation in PostgreSQL
- Problems of changing AQL implementations are similar to those in **migrating an application from one SQL implementation to another**

Un-implemented functions of the query language

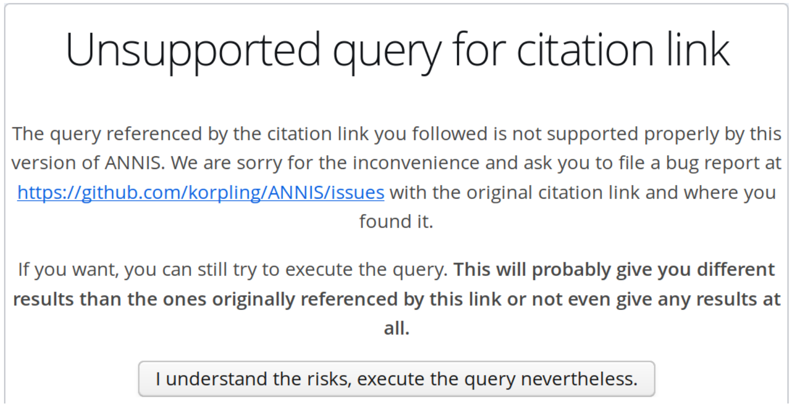
- A query language can accumulate a large number of features over time:
  - AQL has a large number of binary operators that describe the relation between two nodes in the graph
- Orthogonal features can be replaced

Un-implemented functions of the query language

- A query language can accumulate a large number of features over time:
    - AQL has a large number of binary operators that describe the relation between two nodes in the graph
- Orthogonal features can be replaced
- Move implementation of such features to a specific code path that separate the “real” query language from the ugly parts (quirks mode)
- Check stored reference links to make it transparent if a query language feature has actually been used, and therefore needs emulation

Un-implemented functions of the query language

- A query language can accumulate a large number of features over time:
    - AQL has a large number of binary operators that describe the relation between two nodes in the graph
- Orthogonal features can be replaced
- Move implementation of such features to a specific code path that separate the "real" query language from the ugly parts (quirks mode)
- Check stored reference links to make it transparent if a query language feature has actually been used, and therefore needs emulation
- If too hard to implement:


Unsupported query for citation link

The query referenced by the citation link you followed is not supported properly by this version of ANNIS. We are sorry for the inconvenience and ask you to file a bug report at https://github.com/korpling/ANNIS/issues with the original citation link and where you found it.

If you want, you can still try to execute the query. **This will probably give you different results than the ones originally referenced by this link or not even give any results at all.**

Identifiers might change

- Each node in the graph has a URI as an internal identfier
- Matches are lists of URIs and additional names for the label the match refers to
- A match for a query is only the same if all URIs are the same

Identifiers might change

- Each node in the graph has a URI as an internal identfier
- Matches are lists of URIs and additional names for the label the match refers to
- A match for a query is only the same if all URIs are the same

People will use weird names

- Spaces, slashes, umlauts, …
- Double percent-escaped characters

Identifiers might change

- Each node in the graph has a URI as an internal identfier
- Matches are lists of URIs and additional names for the label the match refers to
- A match for a query is only the same if all URIs are the same

People will use weird names

- Spaces, slashes, umlauts, ...
- Double percent-escaped characters
- Everything Unicode has to offer

`To invoke the hive-mind representing chaos.`

https://github.com/minimaxir/big-list-of-naughty-strings

→ Importing data via IDs – and comparing them – is hard

Regular expressions

Regular expressions are an important part of AQL for matching node and edge labels

Regular expressions

Regular expressions are an important part of AQL for matching node and edge labels

```
-- PostgreSQL
SELECT * FROM t WHERE a ~ 'val.*';
-- MySQL
SELECT * FROM t WHERE a REGEXP 'val.*';
```

- Syntax varies from each implementation, even if "supporting POSIX"
- Regular expression engines often allow a search for non-regular expressions, such as backreferences and other extensions: some implementations trade features for speed (e.g RE2 from Google)
- "Power users" will use all regular features available, even if never officially documented

String ordering/collation

For query results, the order of the results is important, e.g. when refering to matches
→ ANNIS 4 reference migration checks order of the matches as well

What is the result of the following SQL query?

```
SELECT '_' < '-';
```

String ordering/collation

For query results, the order of the results is important, e.g. when refering to matches
→ ANNIS 4 reference migration checks order of the matches as well

What is the result of the following SQL query?

```sql
SELECT '_' < '-';
```

- Depends on your localization! `LANG=C` != `LANG=en_US.UTF-8` != `LANG=de_DE.UTF-8`
- PostgreSQL allows to configure the collation for a column of a table explicitly

```sql
CREATE TABLE test1 (a text COLLATE "de_DE");
```

Anyone ever defined their tables this way *before* having collation issues in production?

Conclusion

Conclusion

- ANNIS 4 is currently in public beta
- Users created 12.828 reference links so far on *our* public ANNIS 3 installation alone

Conclusion

- ANNIS 4 is currently in public beta
- Users created 12.828 reference links so far on *our* public ANNIS 3 installation alone

All but 137 queries are known to give the same results in ANNIS 4

Conclusion

- ANNIS 4 is currently in public beta
- Users created 12.828 reference links so far on *our* public ANNIS 3 installation alone

All but 137 queries are known to give the same results in ANNIS 4

- Issues remaining
    - Unsupported regular expression features
    - Unsupported binary operators (might not fix)
    - Actual bugs

Conclusion

- ANNIS 4 is currently in public beta
- Users created 12.828 reference links so far on *our* public ANNIS 3 installation alone

All but 137 queries are known to give the same results in ANNIS 4

- Issues remaining
  - Unsupported regular expression features
  - Unsupported binary operators (might not fix)
  - Actual bugs
- Having these reference links gives us a huge **real world test set**
- **Automatic migration** for persistent IDs
- **Transparency** for the **administrator** which queries she/he can migrate to a new instance
- **Transparency** for the **end-user** if a query is known not to work, no silent failure
- We will be able to **retire ANNIS 3** while keeping all these reference links valid

Appendix

References

Ceri, Stefan, Georg Gottlob, and Letizia Tanca. 1989. "What You Always Wanted to Know About Datalog (and Never Dared to Ask)." *IEEE TRANSACTIONS KNOWLEDGE AND DATA ENGINEERING* 1 (1): 146–66.

Druskat, Stephan, Volker Gast, Thomas Krause, and Florian Zipser. 2016. "Corpus-Tools.org: An Interoperable Generic Software Tool Set for Multi-Layer Linguistic Corpora." In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, edited by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, et al. Portorož, Slovenia: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2016/pdf/918_Paper.pdf.

Krause, Thomas. 2019. "ANNIS: A Graph-Based Query System for Deeply Annotated Text Corpora." Doctoral Dissertation, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät. https://doi.org/10.18452/19659.

Preston-Werner, Tom. n.d. "Semantic Versioning 2.0.0. Semantic Versioning." Accessed May 28, 2019. https://semver.org/.