Piotr Pęzik
piotr.pezik@uni.lodz.pl

Maciej Buczek
mjb@kolos.math.uni.lodz.pl

PELCRA,
University of Łódź

CORPORA as graph databases

Graph databases

- Graph databases are increasingly popular members of the NoSQL (Not Only SQL) database family (along Key-Value stores, Big Table implementations and Document Stores)
- A graph database uses graph structures with nodes (vertices), edges and properties as a means of storing and representing data
- Graph databases scale more naturally than RDBMs for highly connected data since they rely on graph traversals algorithms rather than set operations

Corpora as graphs?

- Corpora have been stored and processed using a variety of general and home-grown solutions
- PELCRA group has used both:
  - RDB-based corpus search engines (good management of complexity, limited performance)
  - IR solutions (Apache Lucene) excellent scalability, difficult to represent structured data (just sentences as token streams)
- Can graph databases be used to store and process corpora (richly) annotated with morphological, syntactic, semantic, structural and bibliographic metadata?

Experiments

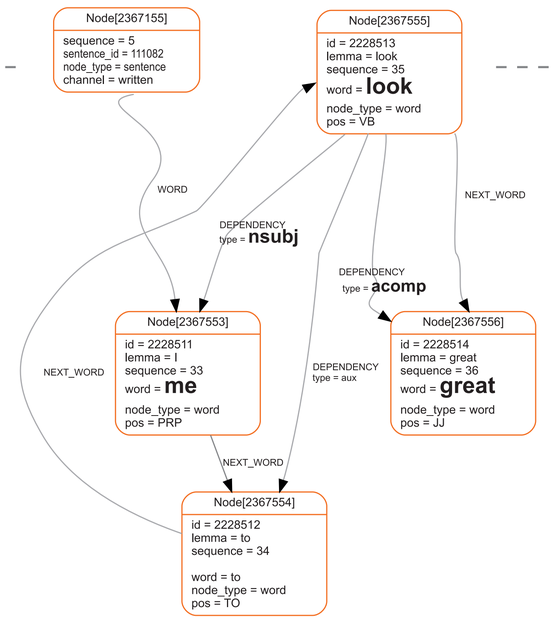
- Reporting on experiments with neo4J (neo4j.org)
- Open-source graph database used to store and process:
  - Linguistically annotated corpora of Polish and English (both monolingual and parallel)
  - Social media text data
  - Phraseological dictionaries
- Very good data representation flexibility
- Good performance, although customized full-text indexing is needed for very large corpora

CASE 1: Lexico-syntactic queries

- Extracting positional and lexicon-syntactic patterns, e.g. Subject-Verb-Complement for a given adjective
- Words, sentences, paragraphs and texts are nodes (vertices) with properties
- Typed edges are used to express word positions and syntactic relations, but also structural and bibliographic annotation
- Cypher query language used to describe sentence-level sub-graphs matching the lexico-syntactic patterns

Cypher query

```
start
w1=node:words(lemma="lonely")
match
w1<-[:NEXT_WORD*1..9]->w3,
w1<-[dep_acomp:DEPENDENCY]-w2-[dep_nsubj:DEPENDENCY]->w3
where
w1.pos =~ /J.*/
and w2.pos =~/V.*/
and (w3.pos =~ /PR.*/ or w3.pos =~/N.*/)
and dep_acomp.type="acomp"
and dep_nsubj.type="nsubj"
return
w3.lemma, w3.pos, w2.lemma, w2.pos, w1.lemma,dep_acomp.type
```

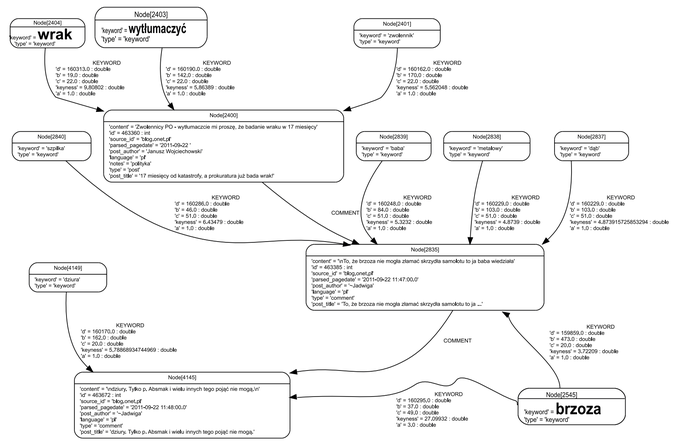


CASE 2: Text-mining in social media

- Social media offer new channels of communication
- Posts, authors, comments are nodes (vertices) with properties
- Full text indexes (Apache Lucene) are used for text content properties of nodes
- Interaction and authorship are represented as edges
- Excellent performance for traversal queries, e.g. finding the longest/shortest conversation path for dozens of degrees of separation

100 longest conversational paths:

```
start post=node:post_properties(type="post")
match
reply_path=post-[:COMMENT*1..100]->comment,
post<-[:KEYWORD]-k_start,
comment<-[:KEYWORD]-k_end
return
length(reply_path) as path_len, k_start.keyword,k_end.keyword
order by path_len desc
limit 100
```

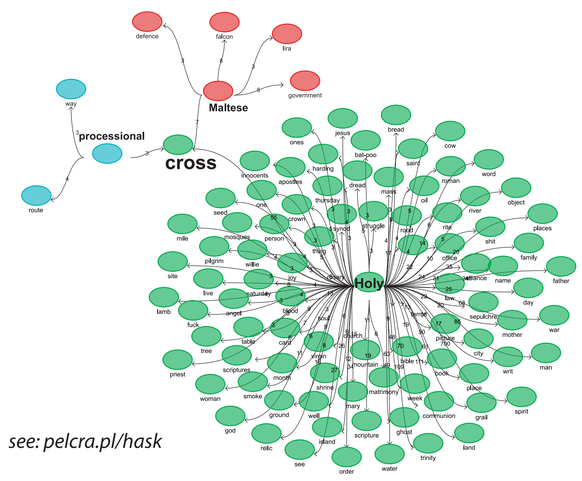


CASE 3: Phraseology and graph theory

- HASK phraseological dictionaries for Polish and English, see pelcra.pl/hask
- Words are represented as vertices, recurrent co-uccurences are edges
- Phraseological productivity is modelled as graph connectivity
- For example we can search for shared collocates using graph traversal patterns

Finding common collocates:

```
start
se1=node:supernodes(node="cross")
match
se1-[:ENTRY]->e1-[col_e1:COLLOCATION]->c1-[:IS_COLLOCATE]->se2
where se2.node_pos="adj"
WITH se1 as se11, se2 as se22, col_e1.TTEST as tt
match
se22<-[:IS_COLLOCATE]-c2<-[col_e2:COLLOCATION]-e2<-[:ENTRY]-se3
where  se3.node_pos="noun"
return se11.node, se22.node, se3.node
```



see: pelcra.pl/hask