GalaTex

W3C XQuery Full-Text Implementation

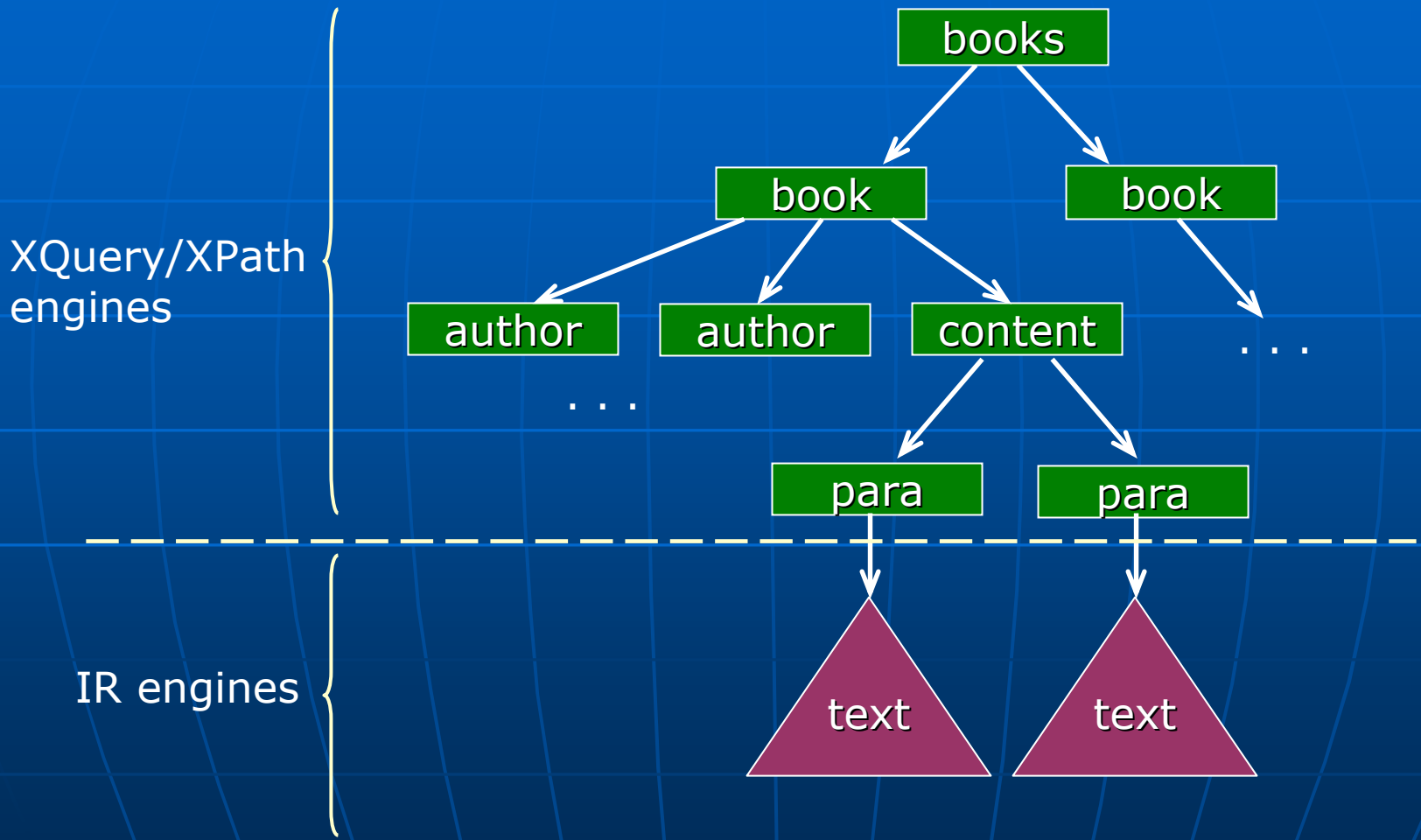
Emiran Curtmola
University of California San Diego

Sihem Amer-Yahia, Philip Brown, Mary Fernández
AT&T Research Labs

Outline of the talk

- Motivation
- Architecture
- Full-text search primitives
- Optimizations
- GalaTex Demo

Motivation


XQuery/XPath engines
IR engines
books
book
book
author
author
content
. . .
. . .
para
para
text
text

Motivation

Querying / Tools	Structure	Text	Scoring results
IR engines (i.e. Google, XXL, JuruXML, Elixir etc.)	Simple path expressions	Word, Boolean, Phrase, Proximity	Powerful
XPath 2.0 XQuery 1.0 XSLT 1.0	Powerful tree manipulation primitives	Substring matching only (i.e. contains, start-with, end-with) + string manipulation functions	None
XQuery Full-Text	Powerful tree manipulation primitives	Fully composable primitives: FTSingleSearch(Token/Phrase), FTWordsSelectionWord(All/Any), FTWordsSelection(All/Any), FTAnd, FTOr, FTNegation, FTMildNegation, FTDistance, FTWindow, FTScope, FTTimes, FTOrdered	Under develop-ment

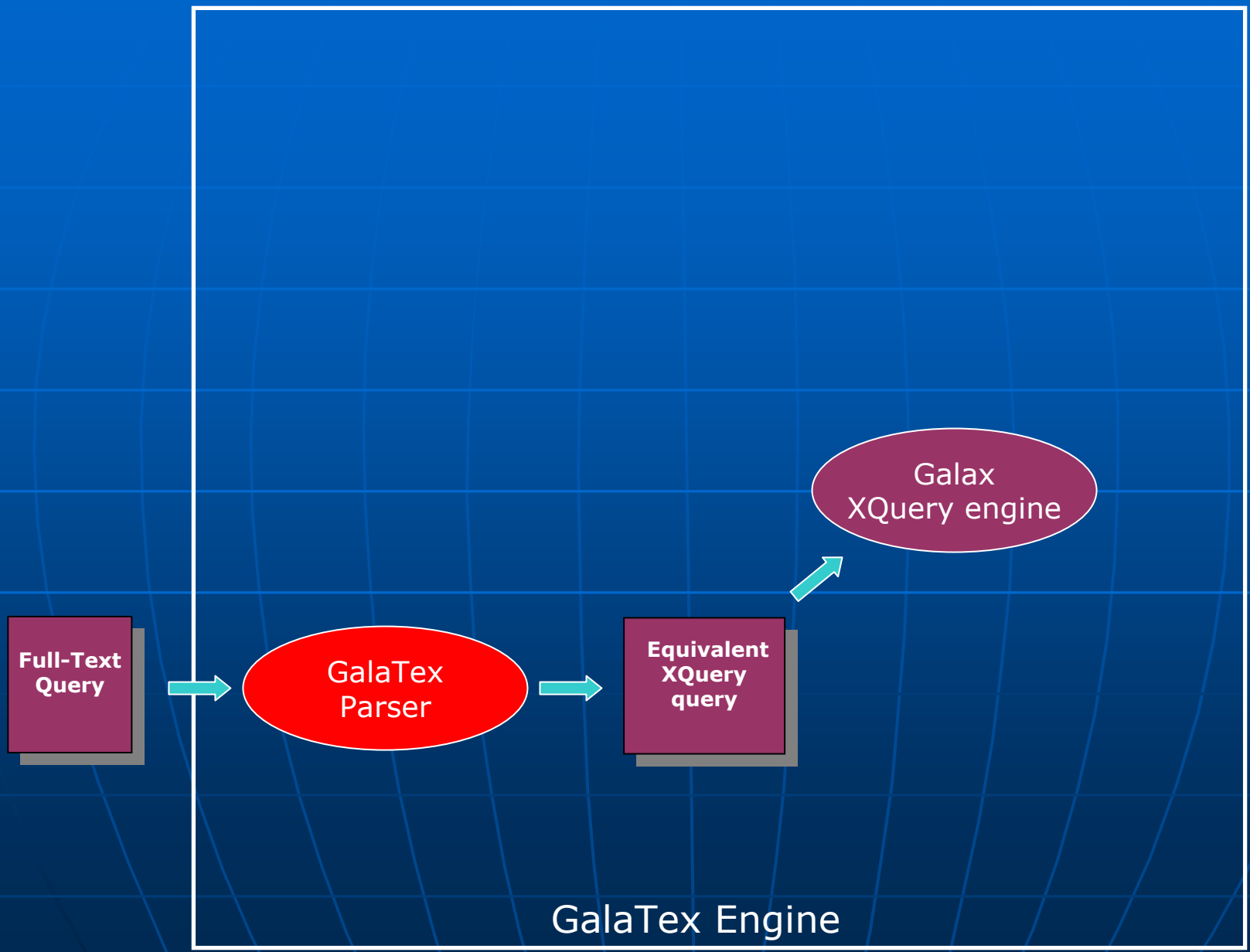
Full-Text Search in XML

- Context expression (evaluation context)
  i.e. set of book chapter nodes
- Return expression
  i.e. book title and paragraph nodes
- Search expression
  - full-text search primitives: and, distance…
- Score expression
  - scoring and ranking the results

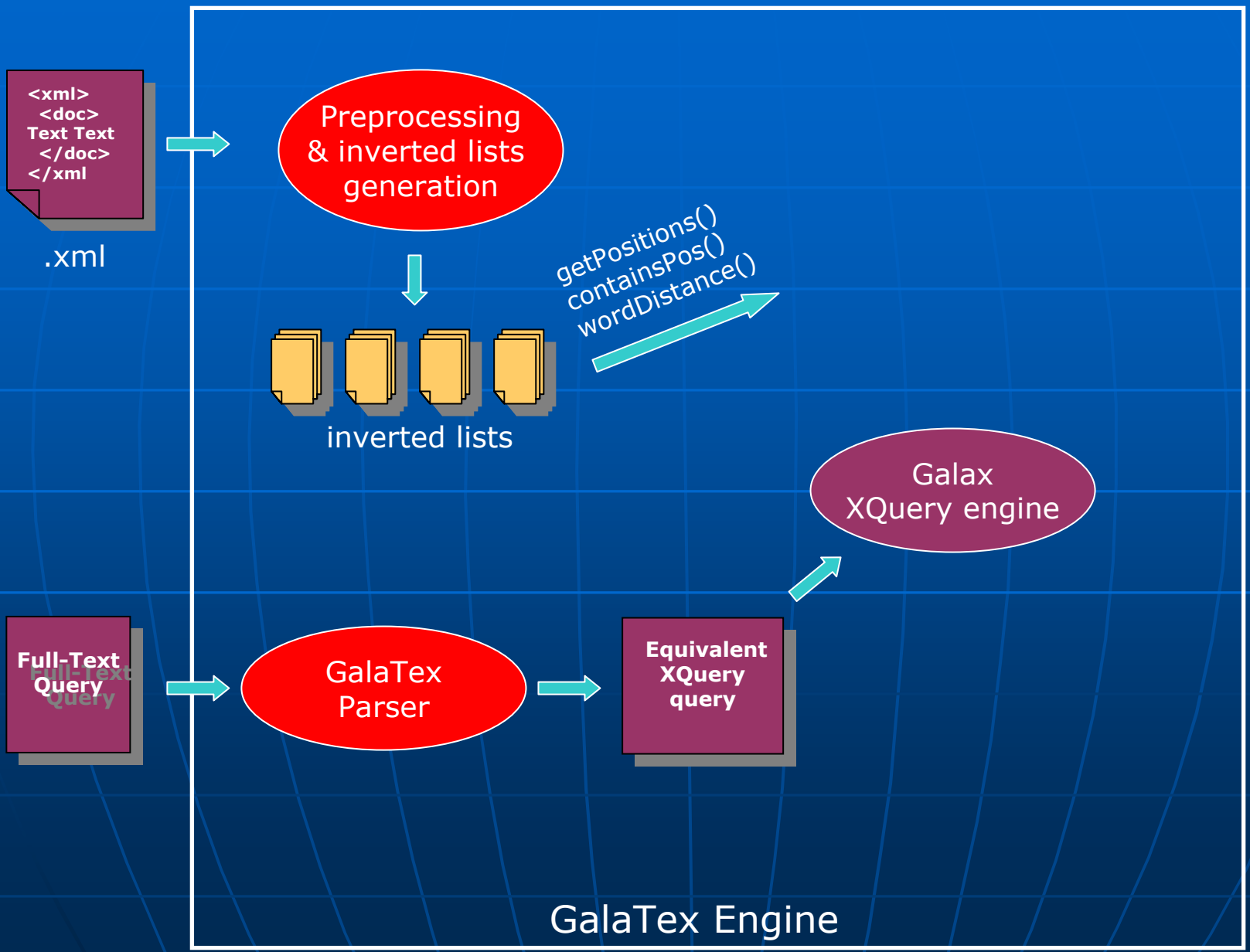
The Goal

- Recommended W3C standards for XML querying: XPath 2.0 and XQuery 1.0
- Galax – fully compliant implementation of XPath 2.0/XQuery 1.0 developed by AT&T Labs (Mary Fernández) & Bell Labs
- W3C XQuery Full-Text spec. (first public draft in July 2004) based on the TeXQuery language (Amer-Yahia et al. WWW'04)
- GalaTex = **Galax + Full-Text search** ***integration of queries on structured data and text data***

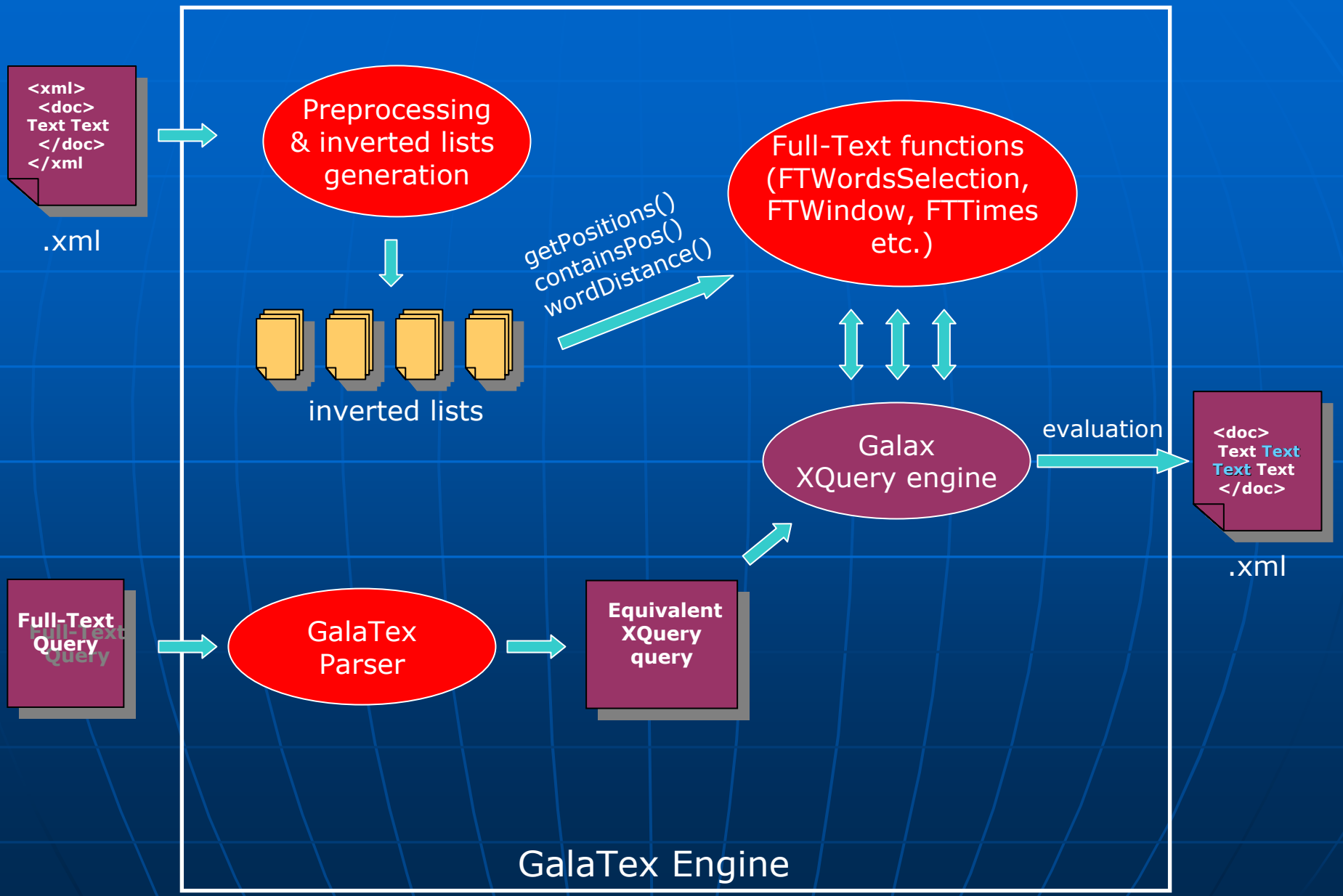
Architecture



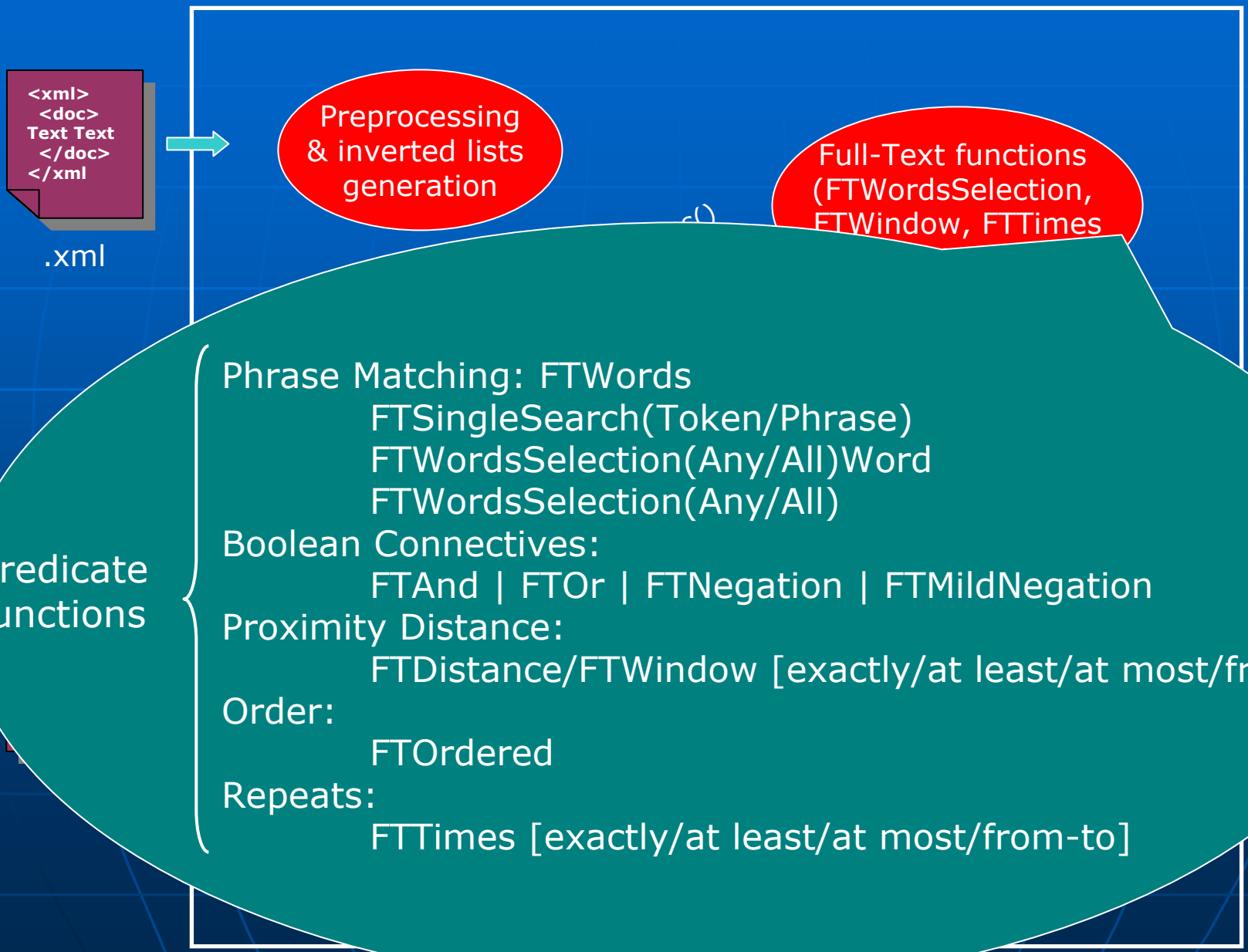
Architecture



Architecture

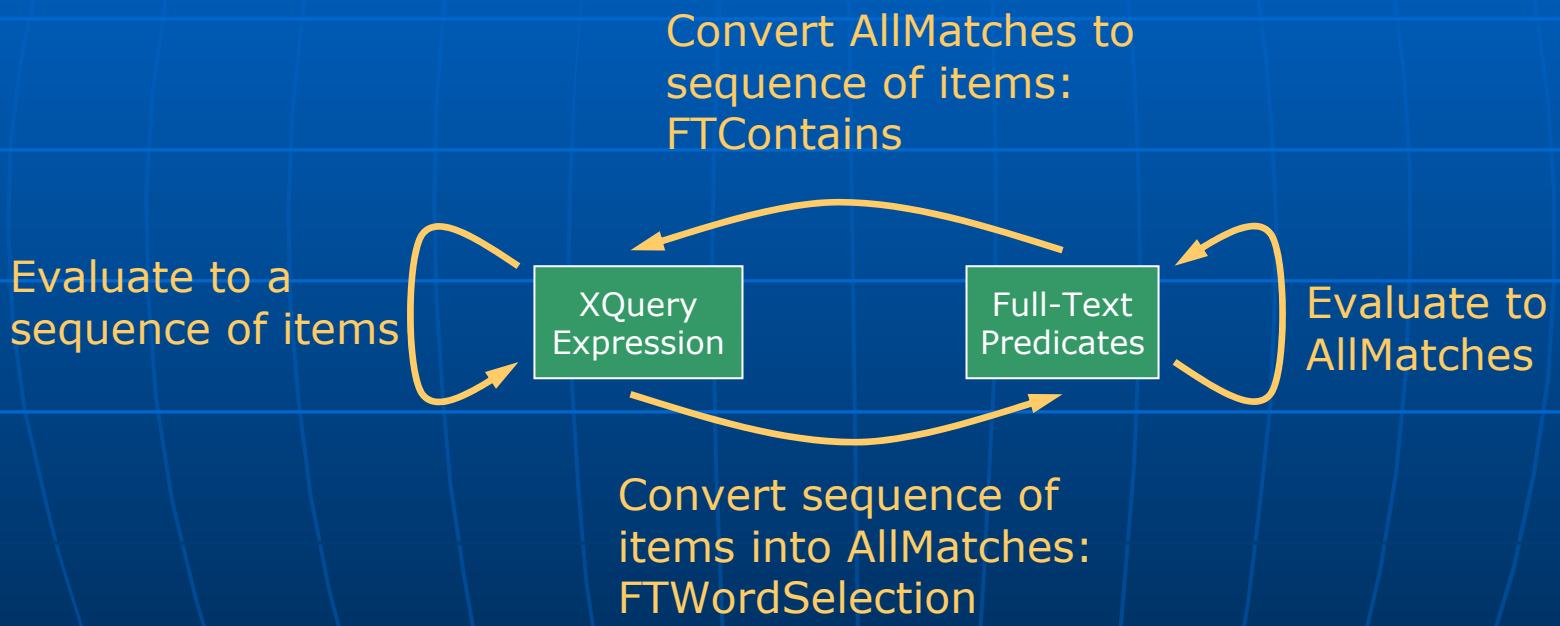


Architecture



Composability

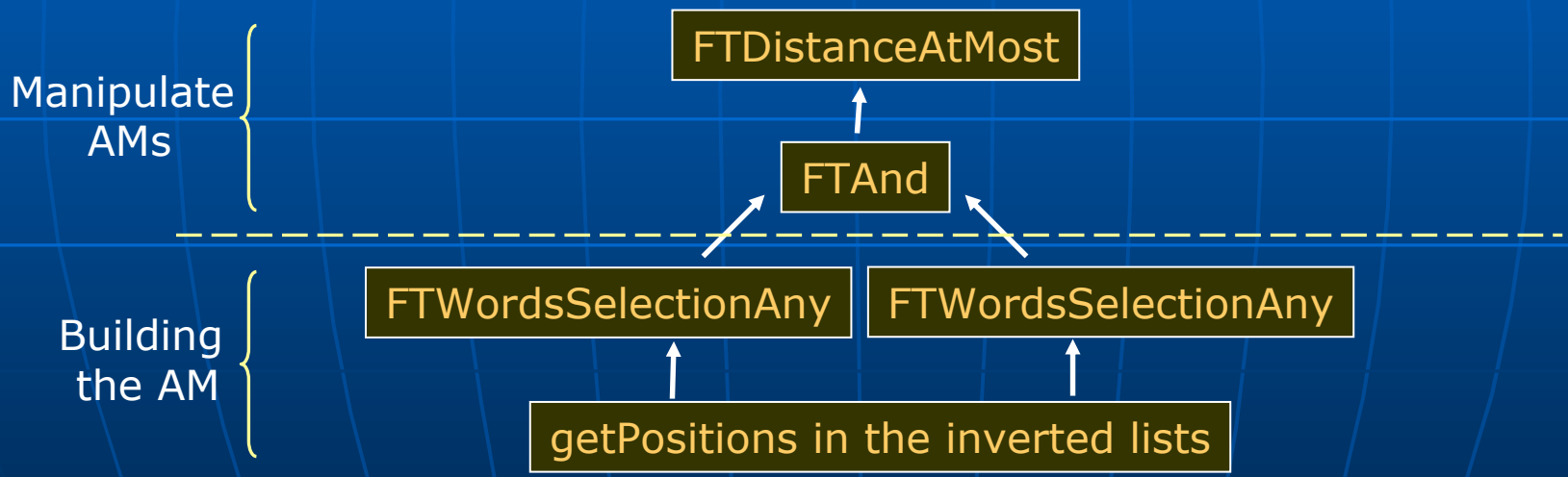
- Co-languages: Galax and GalaTex



Composability

- Full-text predicates

  Example query = *find all book paragraphs that contain "software" and "users" at a distance at most 13 words of each other*



FT Predicates – example

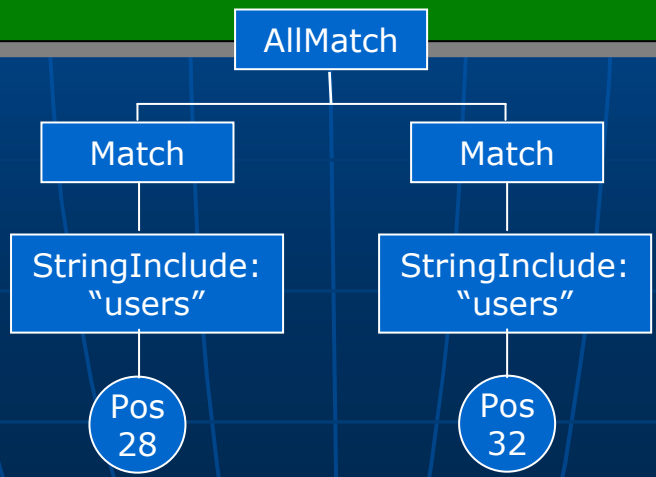
- GalaTex data model: AllMatches
  - positions of tokens within XML nodes
  - all possible matches
  - XML representation
- Input query: all "users" in book contents

```
<book(1) id(2)="1000(3)">
  <author(4)>Mary(5) Rose(6)</author(7)>
  <content(8)>
    <p(9)> The(10) usability(11) of(12) software(13) measure(14) how(15) well(16) the(17)
           software(18) provides(19) support(20) for(21) quickly(22) achieving(23) specified(24)
           goals(25) for(26) the(27) users(28). </p(29)>
    <p(30)> The(31) users(32) must(33) be(34) and(35) feel(36) well-served(37).</p(38)>
  </content(39)>
</book(40)>
```

FT Predicates – example

- GalaTex data model: AllMatches
  - positions of tokens within XML nodes
  - all possible matches
  - XML representation
- Input query: all "users" in book contents

```
<book(1) id(2)="1000(3)">
  <author(4)>Mary(5) Rose(6)</author(7)>
  <content(8)>
    <p(9)> The(10) usability(11) of(12) software(13) measure(14) how(15) well(16) the(17)
           software(18) provides(19) support(20) for(21) quickly(22) achieving(23) specified(24)
           goals(25) for(26) the(27) users(28). </p(29)>
    <p(30)> The(31) users(32) must(33) be(34) and(35) feel(36) well-served(37).</p(38)>
  </content(39)>
</book(40)>
```



FT Primitives – example

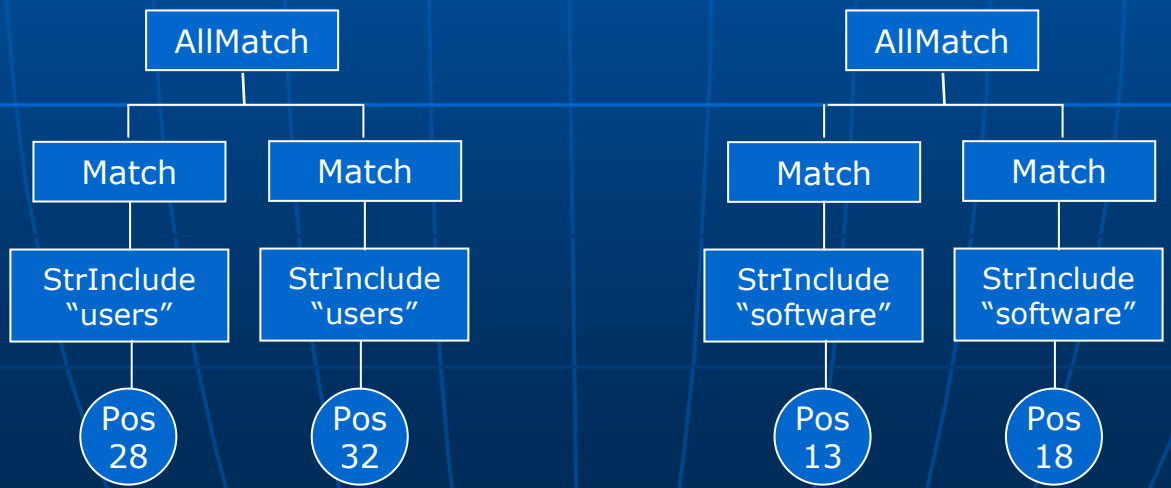
- FTAnd + FTDistance: *//book/content/p[. ftcontains "users" && "software" at distance at most 13 words]*

```
<book(1) id(2)="1000(3)">
  <author(4)>Mary(5) Rose(6)</author(7)>
  <content(8)>
    <p(9)> The(10) usability(11) of(12) software(13) measure(14) how(15) well(16) the(17)
           software(18) provides(19) support(20) for(21) quickly(22) achieving(23) specified(24)
           goals(25) for(26) the(27) users(28). </p(29)>
    <p(30)> The(31) users(32) must(33) be(34) and(35) feel(36) well-served(37).</p(38)>
  </content(39)>
</book(40)>
```

FT Primitives – example

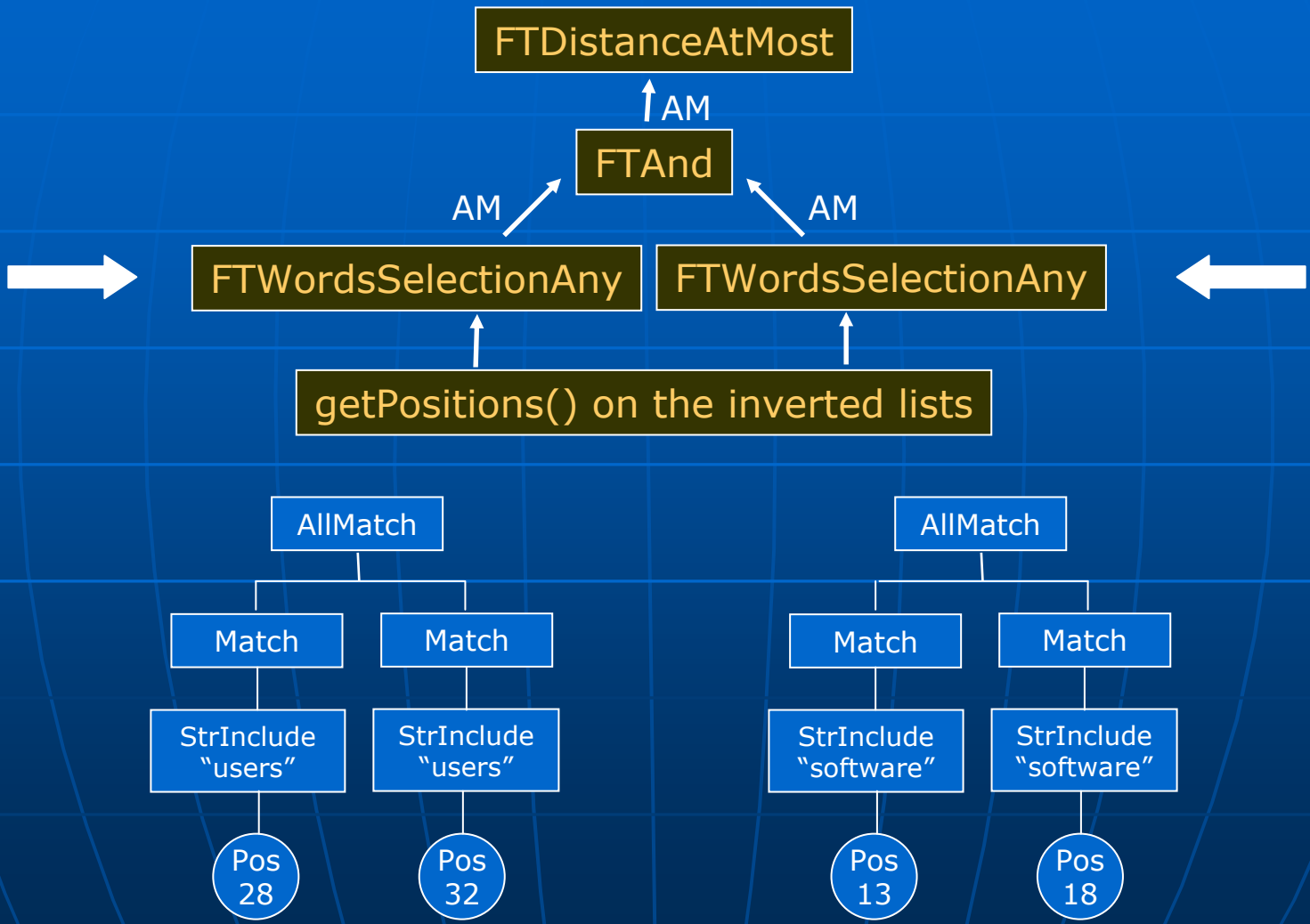
- FTAnd + FTDistance: *//book/content/p[. ftcontains "users" && "software" at distance at most 13 words]*

```
<book(1) id(2)="1000(3)">
  <author(4)>Mary(5) Rose(6)</author(7)>
  <content(8)>
    <p(9)> The(10) usability(11) of(12) software(13) measure(14) how(15) well(16) the(17)
           software(18) provides(19) support(20) for(21) quickly(22) achieving(23) specified(24)
           goals(25) for(26) the(27) users(28). </p(29)>
    <p(30)> The(31) users(32) must(33) be(34) and(35) feel(36) well-served(37).</p(38)>
  </content(39)>
</book(40)>
```



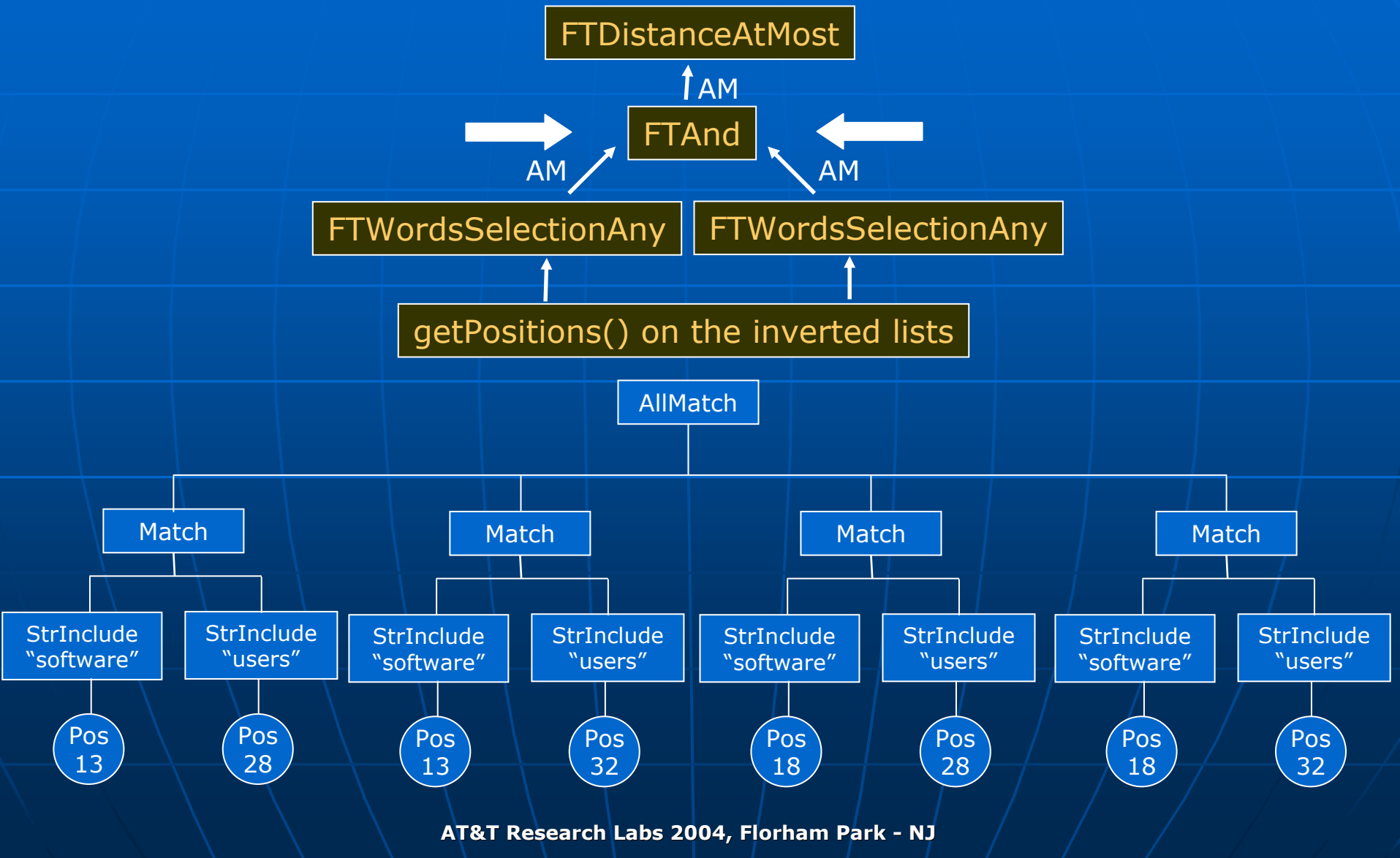
FT Primitives – example

- FTAnd + FTDistance: *//book/content/p[. ftcontains "users" && "software" at distance at most 13 words]*



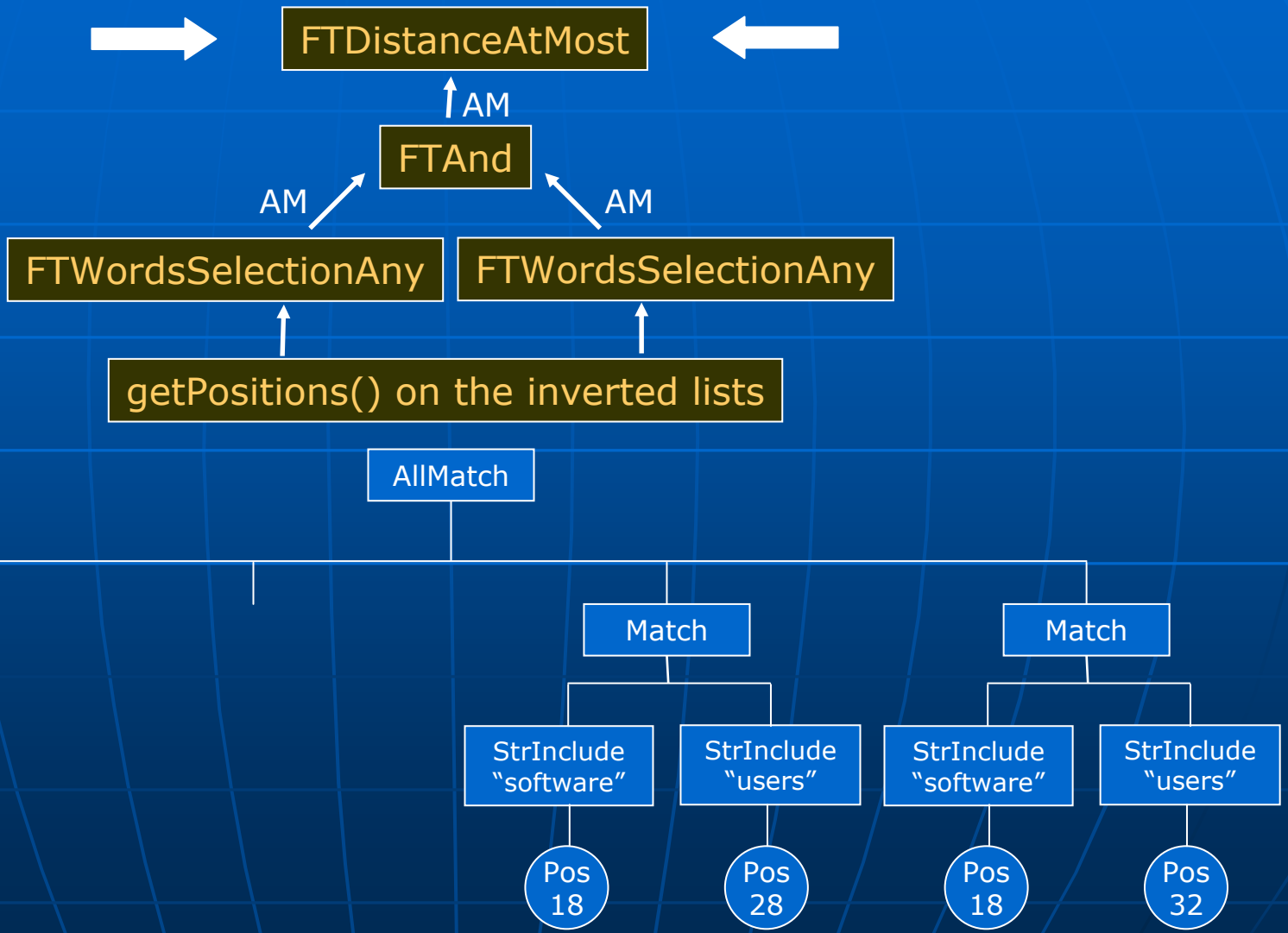
FT Primitives – example

- FTAnd + FTDistance: *//book/content/p[. ftcontains "users" && "software" at distance at most 13 words]*



FT Primitives – example

- FTAnd + FTDistance: *//book/content/p[. ftcontains "users" && "software" at distance at most 13 words]*

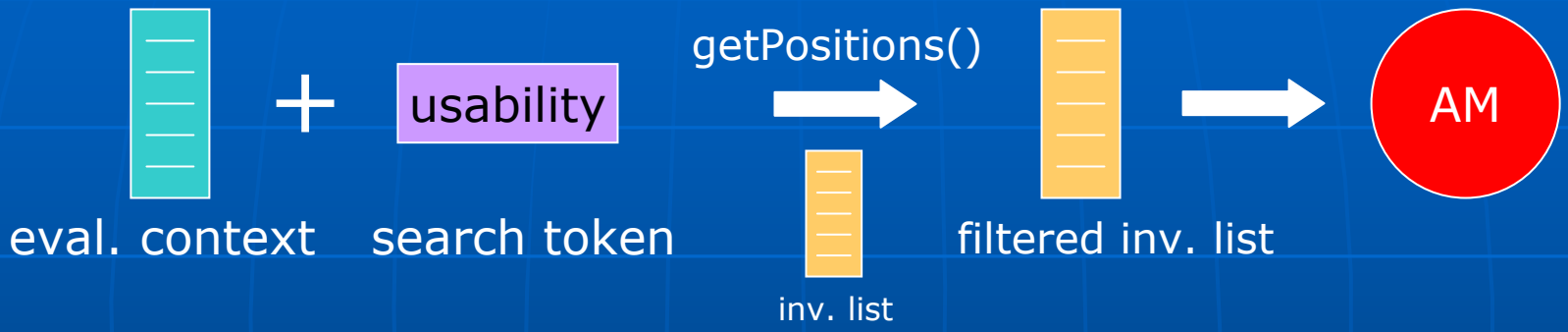


Match Options

- Applicable on any Full-text predicate
  - Case sensitive (FTCaseOption)
  - Regular expression (FTRegexOption)
  - Stop words (FTStopwordOption in FTDistance/FTWindow)
  - Special chars (FTSpecialcharOption)
  - Stemming - word variations (FTStemOption)
  - Ignore (FTIgnoreOption - PIX)
  - Thesaurus (FTThesaurusOption)
  - Lang. diacritics (FTDiacriticsOption)
  - Language (FTLanguageOption)

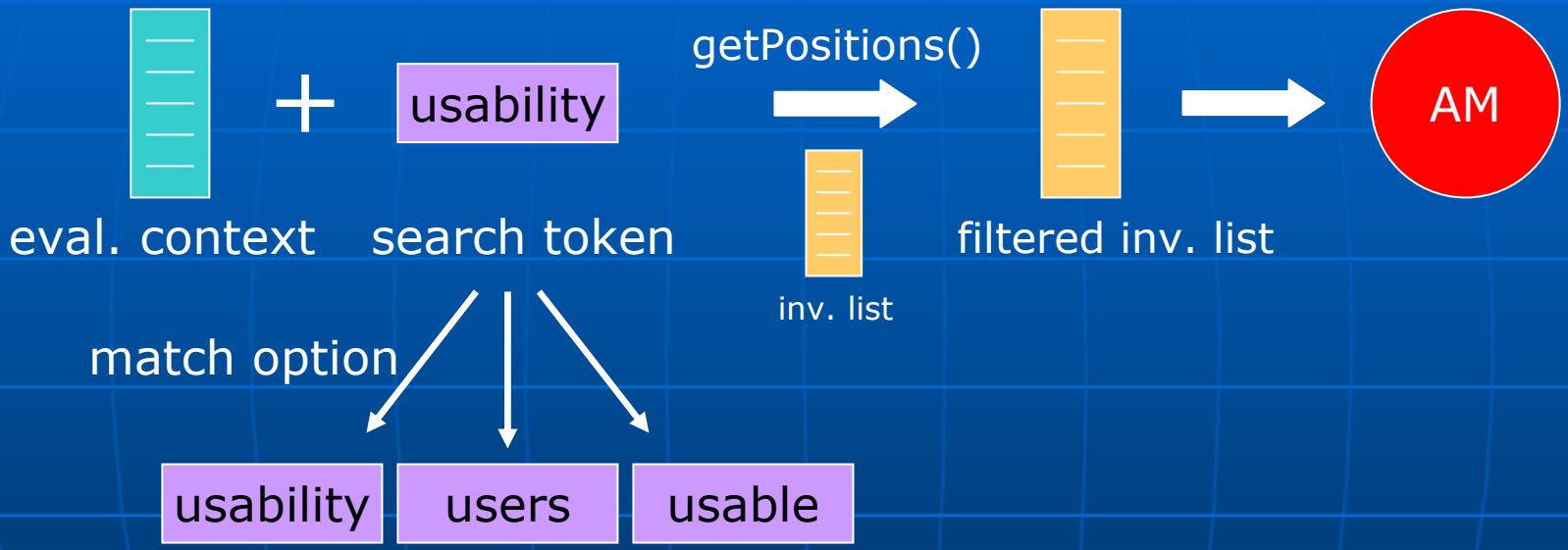
Match Options

Query: //book/content/p[. ftcontains “usability” with stemming]



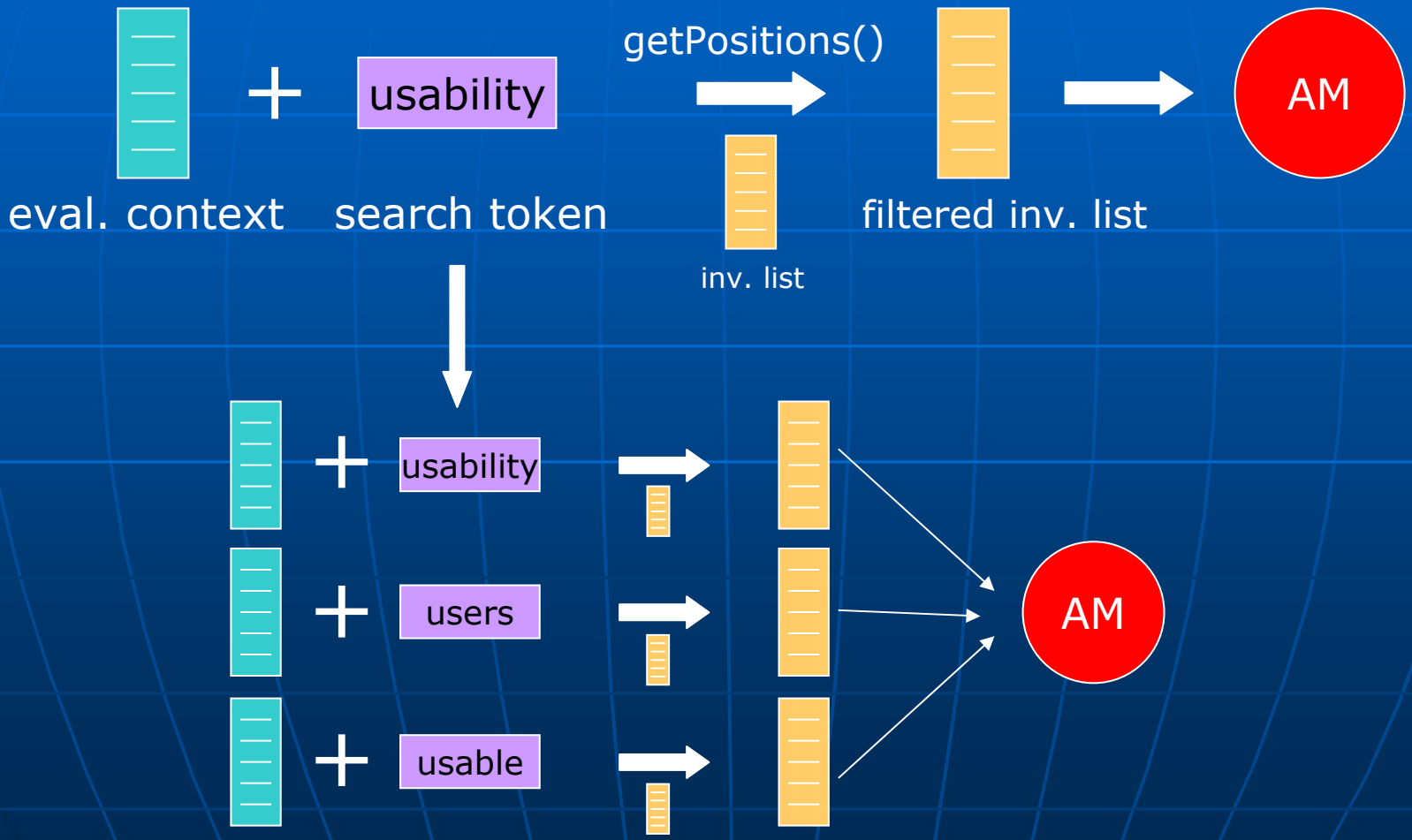
Match Options

Query: //book/content/p[. ftcontains “usability” with stemming]



Match Options

Query: //book/content/p[. ftcontains "usability" with stemming]



Outline of the talk

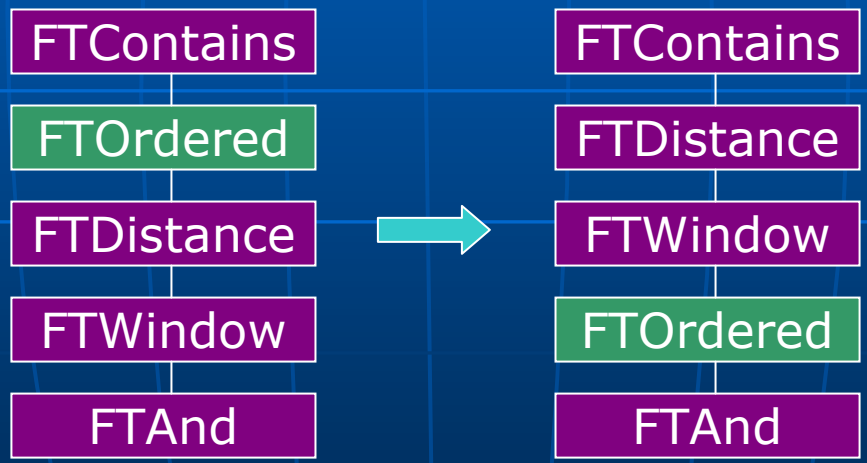
- Motivation
- Architecture
- Full-text search primitives
- Optimizations
- GalaTex Demo

Optimizations

- Node & position lists are a bottleneck
  - Logical rewritings on the primitives tree
    - pushing down more restrictive selections
    - rewriting FTOr into an XQuery “Or”

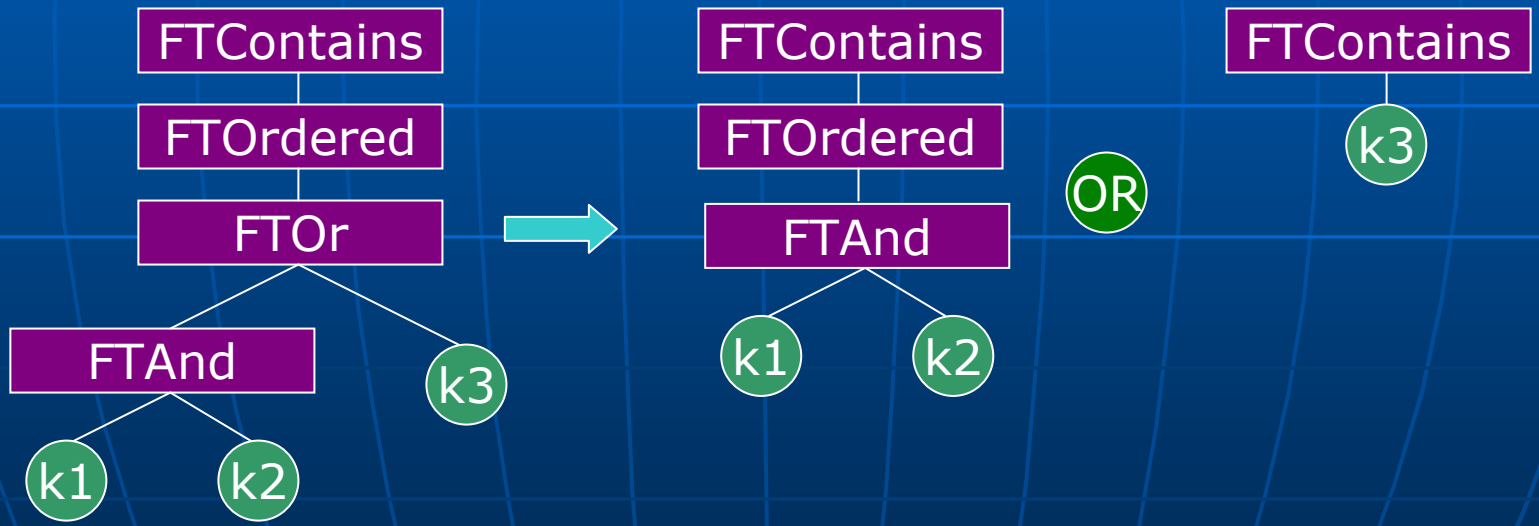
Optimizations

- Logical GalaTex evaluation tree rewrites
  - Pushing down more restrictive selections



Optimizations

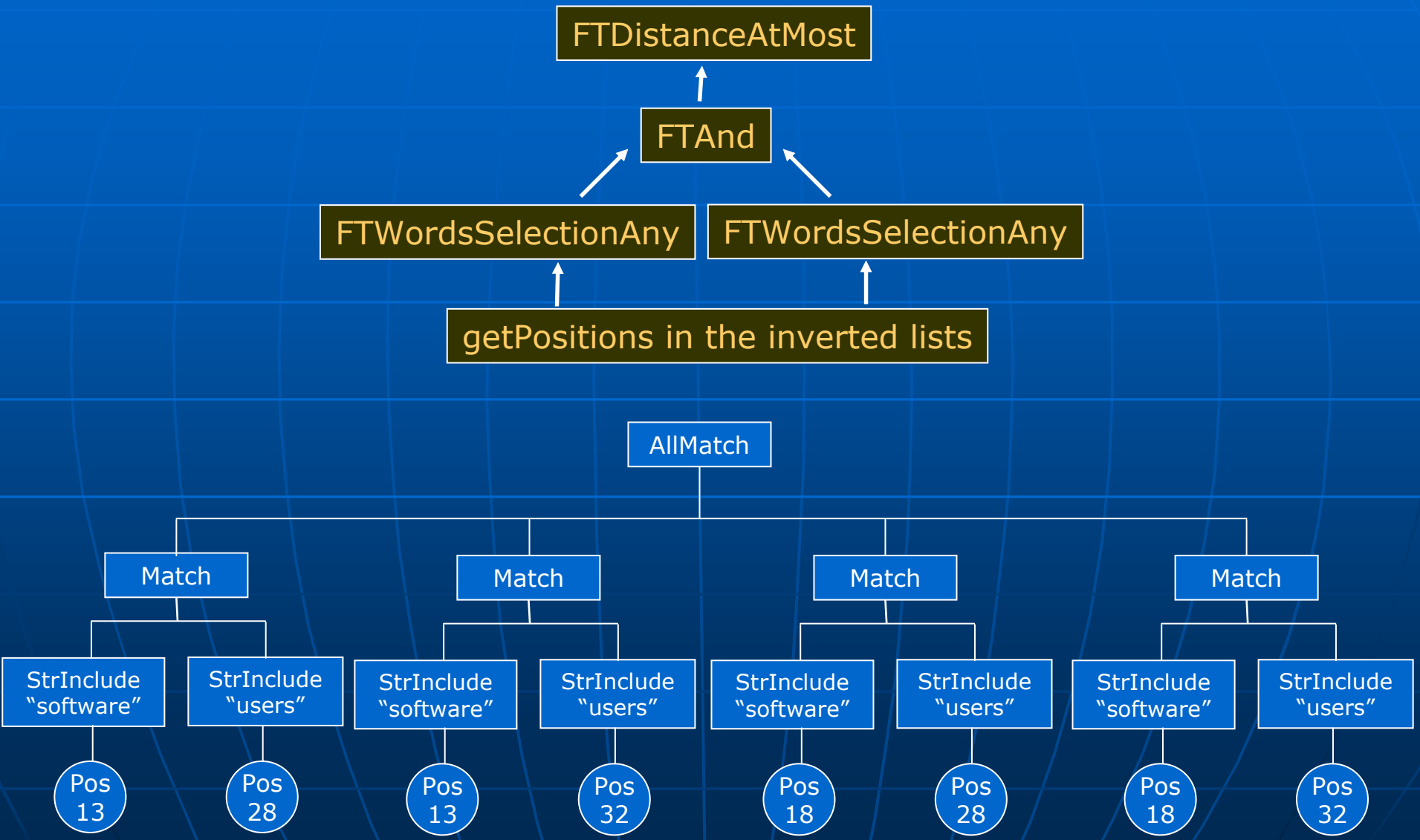
- Logical GalaTex evaluation tree rewrites
  - Spliting an FTOr into an XQuery "OR"



Optimizations

- Node & position lists are a bottleneck
  - Logical rewritings on the primitives tree
    - pushing down more restrictive selections
    - rewriting FTOr into an XQuery “Or”
  - Checking ancestor/descendant relationships between nodes, i.e. pruning EC on LCA
  - Sort merge to filter the inverted lists
    - EC and inverted lists
  - Pipeline evaluation engine
- Scoring: still preserve the above

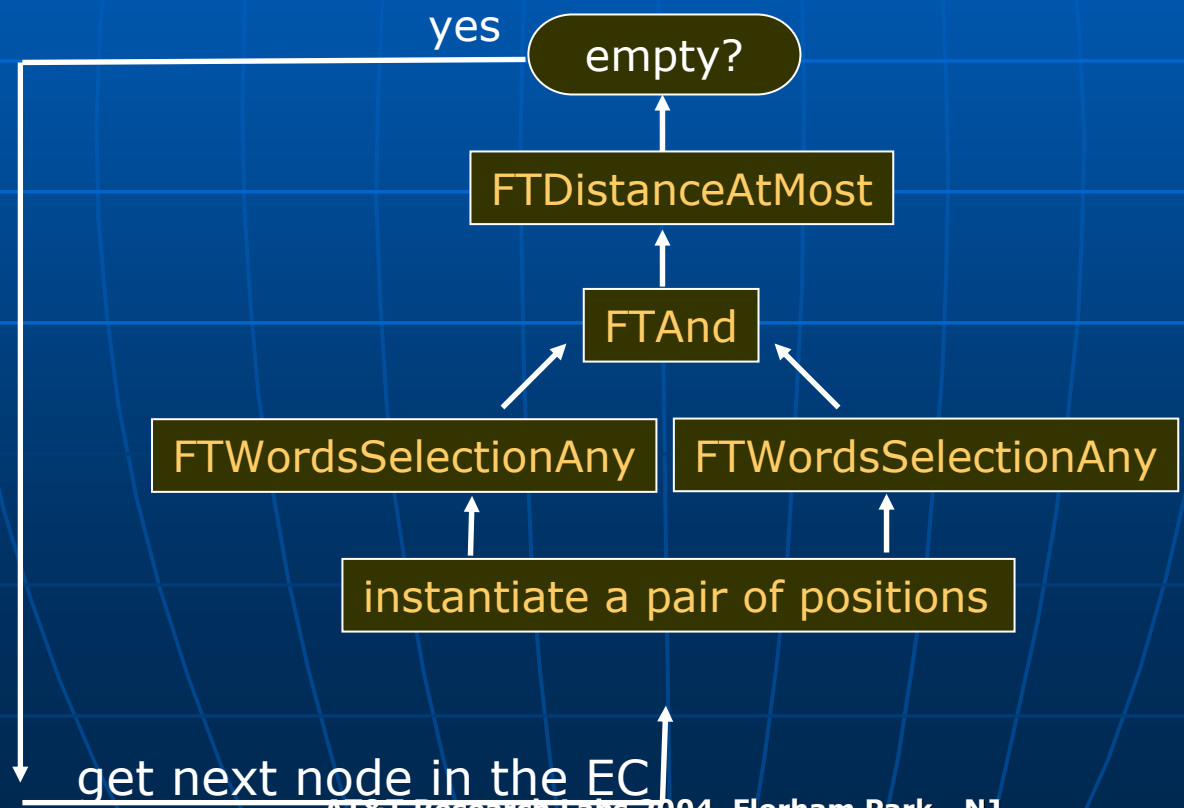
Pipelining


FTDistanceAtMost
FTAnd
FTWordsSelectionAny
FTWordsSelectionAny
getPositions in the inverted lists
AllMatch
Match
Match
Match
Match
StrInclude "software"
StrInclude "users"
StrInclude "software"
StrInclude "users"
StrInclude "software"
StrInclude "users"
StrInclude "software"
StrInclude "users"
Pos 13
Pos 28
Pos 13
Pos 32
Pos 18
Pos 28
Pos 18
Pos 32

Pipelining (Quark)

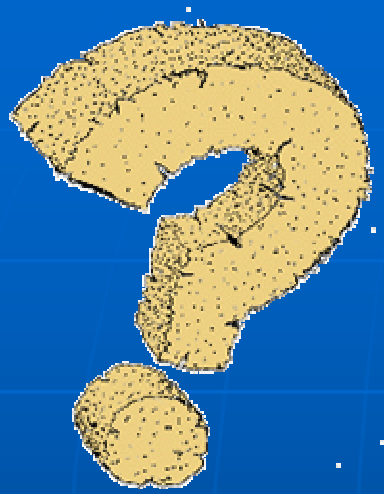
- Inverted lists per token structured as:
- Not materializing
  - Inverted lists
  - Evaluation context

nodeID	list of positions
…	…
…	…



Conclusions

- Demo available at http://www.galaxquery.com/galatex
- Status
  - Inverted lists (document preprocessing)
  - Composable full-text search primitives
  - XML validation (AllMatches data model)
    - element names and types
  - W3C XQuery Full-Text use cases
- Ongoing work
  - Integrate match options and optimizations
  - Performance evaluation
  - Full integration of GalaTex into Galax



Thank you