

Information Retrieval in Joost

Jörg Tiedemann

`tiedeman@let.rug.nl`

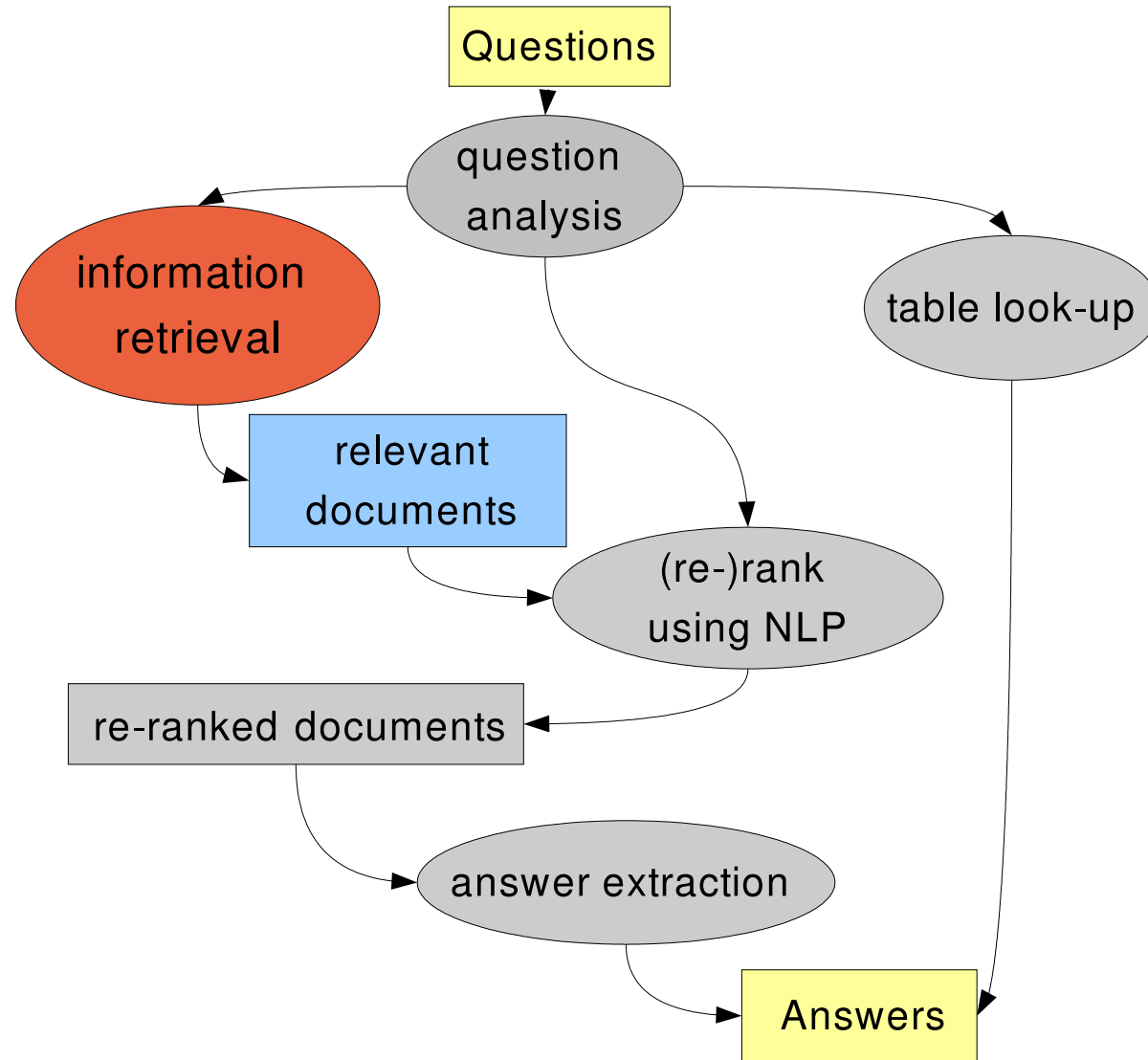
Information Science

Rijksuniversiteit Groningen

Outline

- IR in Joost
- Selecting IR engines
- Integration in Joost
- Playing with parameters
- Integrating NLP in IR

IR in Joost



Questions about IR

- What should we retrieve?
 - documents, paragraphs, sentences
- How much should we retrieve?
 - ... to maximise recall with reasonable precision
- Which IR engine should we use?
 - off-the shelf, customised
- What kind of queries should we use?
 - ... to improve recall and relevance ranking

Initial experiments (using CLEF2003)

● IR: Managing Gigabytes (MG)

Docs			Pars			Sents		
N	Words	R	N	Words	R	N	Words	R
5	2.500	55.7	50	3.325	68.7	100	1.760	62.2
10	5.000	63.2	100	6.650	76.0	200	3.520	68.9
15	7.500	66.5	150	9.975	79.2	300	5.280	72.3
20	10.000	69.2	200	13.300	80.0	400	7.040	74.9
						500	8.800	76.2
						600	10.560	77.6

N = # of doc/par/sent, Words = # words, R = recall

IR engines

- from academia:

Managing Gigabytes (MG): <http://www.cs.mu.oz.au/mg/>

Zettair: <http://www.seg.rmit.edu.au/zettair/>

- programming libraries:

Lucene: <http://jakarta.apache.org/lucene/docs>

Xapian: <http://www.xapian.org/>

- other open source systems:

Amberfish: <http://www.etymon.com/tr.html>

Swish-e: <http://swish-e.org/>

Zebra: <http://www.indexdata.dk/zebra/>

Experiments with CLEF 2003

188,651	documents	● 450 questions
1,101,790	paragraphs	● 370 with answers
4,039,614	sentences	
76,692,515	words	

- evaluate IR ranking using MRR
- evaluate IR recall
- evaluate QA using Joost

Experiments with CLEF 2003

Evaluate by means of MRR

$$MRR = \frac{1}{x} \sum_x \frac{1}{rank(first_answer)}$$

doc MRR: mean reciprocal rank of relevant documents retrieved; i.e. documents listed in the gold standard

answer MRR: mean reciprocal rank of relevant answers retrieved, i.e. documents including the answer string

Experiments with CLEF 2003 (IR)

MRR (in %)	documents		paragraphs		sentences	
	doc	answer	doc	answer	doc	answer
Xapian	28.25	50.49	32.98	44.34	25.14	28.90
Swish-e	26.02	54.01	29.22	43.38	23.85	32.87
Zettair	32.10	52.69	30.61	42.40	28.32	31.04
Lucene	29.74	47.87	32.72	40.25	27.82	29.61
Zebra	26.50	45.06	30.70	39.34	25.47	30.67
Amberfish	21.05	44.31	18.35	26.97	21.15	23.06
MG	20.86	39.98	21.27	22.87	21.18	15.44

Experiments with CLEF 2003 (IR)

number of paragraphs required to obtain $\geq x\%$ recall (max 200 par):

recall	75	76	77	78	79	80	81	82	83	84
Swish-e	34	37	42	59	63	73	89	117	147	-
Zettair	36	40	43	52	70	84	103	140	181	-
Lucene	55	61	67	78	88	98	118	144	-	-
Xapian	42	48	50	61	84	132	140	-	-	-
Zebra	93	115	143	172	193	-	-	-	-	-
MG	109	121	129	137	180	-	-	-	-	-
Amberfish	145	197	-	-	-	-	-	-	-	-

Experiments with CLEF 2003 (QA)

	MRR (paragraphs)	
	<i>strict</i>	<i>lenient</i>
Xapian	0.392	0.559
Lucene	0.399	0.556
Zettair	0.387	0.530
Zebra	0.342	0.501
Swish-e	0.325	0.438
MG	0.297	0.415
Amberfish	0.231	0.304

So what?

- segmentation level: paragraphs
- selected IR engines: Xapian, Zettair, Lucene

Zettair: very fast in indexing, summary function, but small user community (no extensions, no forum ...)

Xapian: proximity queries, blind relevance feedback, wide range of query operators

Lucene: multiple fields, proximity queries, keyword boosting, very popular

And what now?

- IR combination (voting)
- include phrases (multi-word NEs)
- query expansion (EWN synonyms)
- NLP and query optimisation

And what now?

- IR combination (voting) ... **didn't work**
- include phrases (multi-word NEs)
- query expansion (EWN synonyms)
- NLP and query optimisation

And what now?

- IR combination (voting) ... **didn't work**
- include phrases (multi-word NEs) ... **didn't work**
- query expansion (EWN synonyms)
- NLP and query optimisation

And what now?

- IR combination (voting) ... **didn't work**
- include phrases (multi-word NEs) ... **didn't work**
- query expansion (EWN synonyms) ... **didn't work**
- NLP and query optimisation

And what now?

- IR combination (voting) ... **didn't work**
- include phrases (multi-word NEs) ... **didn't work**
- query expansion (EWN synonyms) ... **didn't work**
- NLP and query optimisation ... **let's have a look**

NLP in IR

- basic idea:

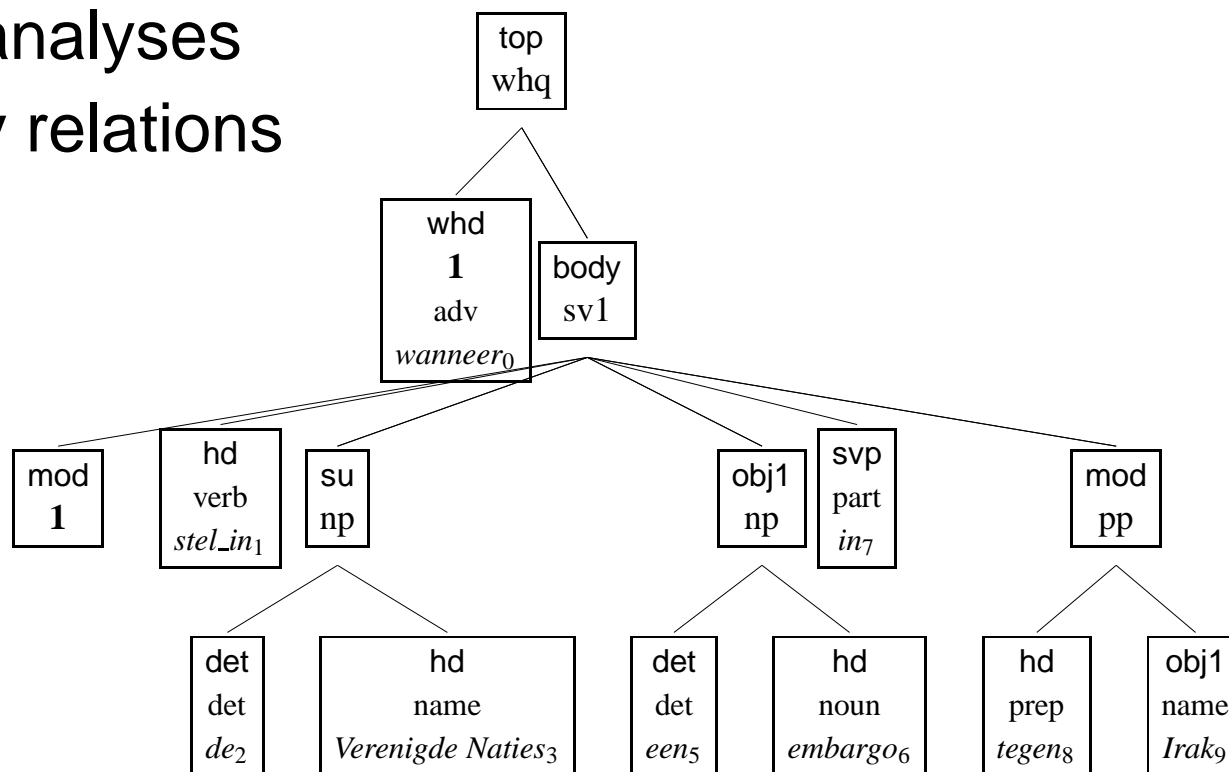
- use NLP for pre-processing (corpus&query)
- include linguistic features in the IR database
- search for keywords and their features in the database

- query optimisation:

- include only those features that help to improve IR
- boost certain feature types compared to others
- NLP can help keyword selection/weighting (e.g. boost 'nouns')

Available annotation

- CLEF corpus is parsed by Alpino (RSI corpus as well)
 - root forms
 - POS
 - named entities (ORG, PER, LOC)
 - compound analyses
 - dependency relations

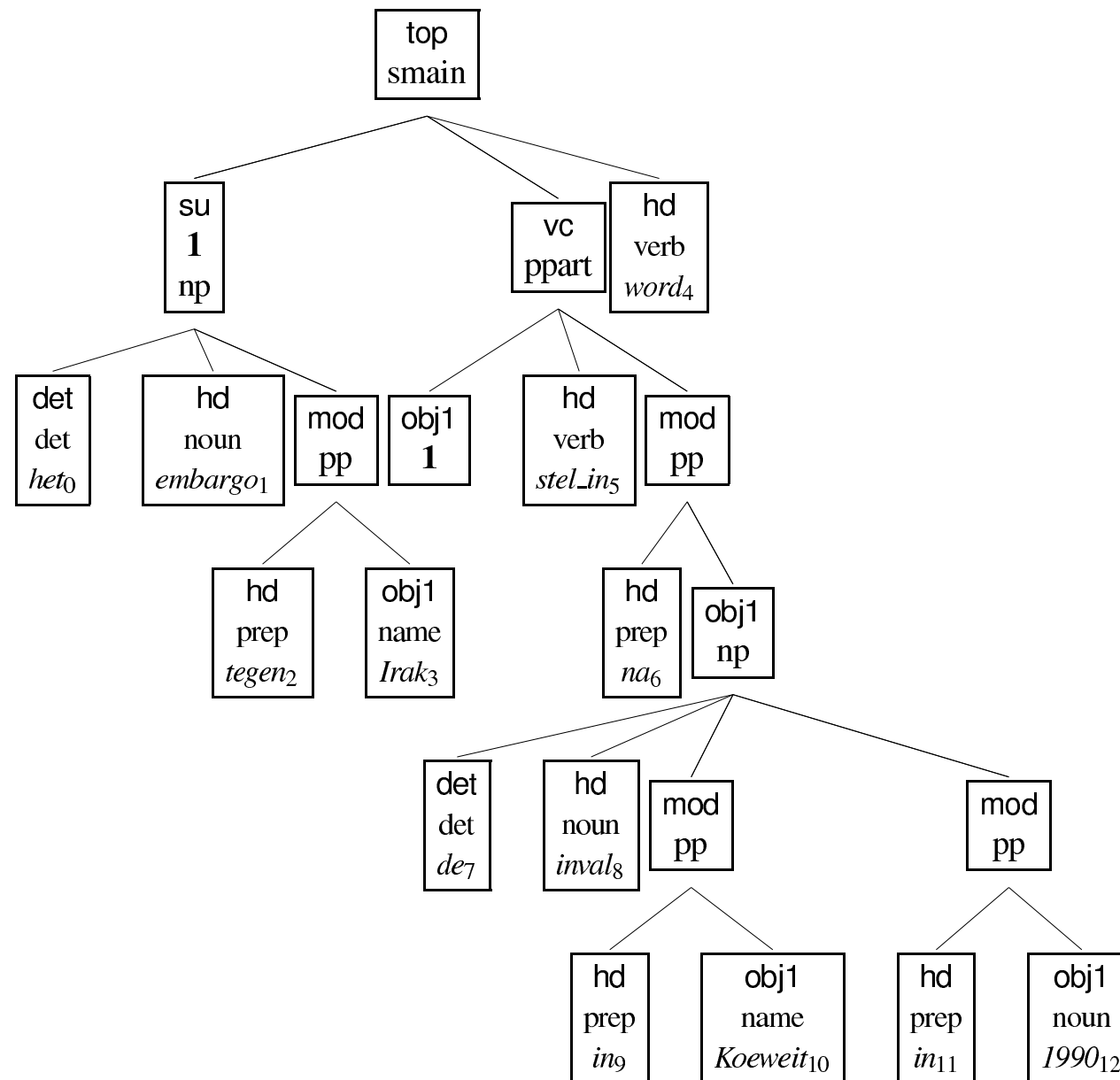


Multi-layer index using Lucene

- store different types of data in different index fields (= layers)

text	plain text (stemmed, compounds split at '–')
root	root-forms (compounds split at '–' and '_')
RootPos	root-form + POS tag
RootHead	root-form + head
RootRel	root-form + relation name
RootRelHead	root-form + relation name + head
compound	compounds (non-split root-forms)
ne/neLOC/nePER/neORG	NEs (root forms, split & non-split)
neTypes	NE labels (ORG, PER, LOC, TMP, YEAR, MEASURE, SCORE)

Het embargo tegen Irak werd ingesteld na de inval in Koeweit in 1990.



Het embargo tegen Irak werd ingesteld na de inval in Koeweit in 1990.

layer	contents
text	Het embargo tegen Irak werd ingesteld na de inval in Koeweit in 1990
root	het embargo tegen Irak word stel in na de inval in Koeweit in 1990
RootPOS	het/det embargo/noun tegen/prep Irak/name word/verb stel_in/verb na/prep de/det inval/noun in/prep Koeweit/name in/prep 1990/noun
RootHead	het/embargo embargo/word tegen/embargo Irak/tegen word/ stel_in/word na/stel_in de/inval inval/na in/inval Koeweit/in in/inval 1990/in
RootRel	het/det embargo/su tegen/mod Irak/obj1 word/ stel_in/vc na/mod de/det inval/obj1 in/mod Koeweit/obj1 in/mod 1990/obj1
RootRelHead	het/det/embargo embargo/su/word tegen/mod/embargo Irak/obj1/tegen word// stel_in/vc/word na/mod/stel_in de/det/inval inval/obj1/na in/mod/inval Koeweit/obj1/in in/mod/inval 1990/obj1/in
compound	stel_in
ne	Irak Koeweit
neLOC	Irak Koeweit
nePER	
neORG	
neTypes	LOC YEAR

Query formulation

Keyword types:

basic: a keyword in one of the index layers

restricted: *token-layer* keywords can be restricted to a certain word class and/or a certain relation type.

Our word class restrictions: **noun, name, adjective, verb**

Relation type restrictions: **direct object, modifier, apposition** and **subject**

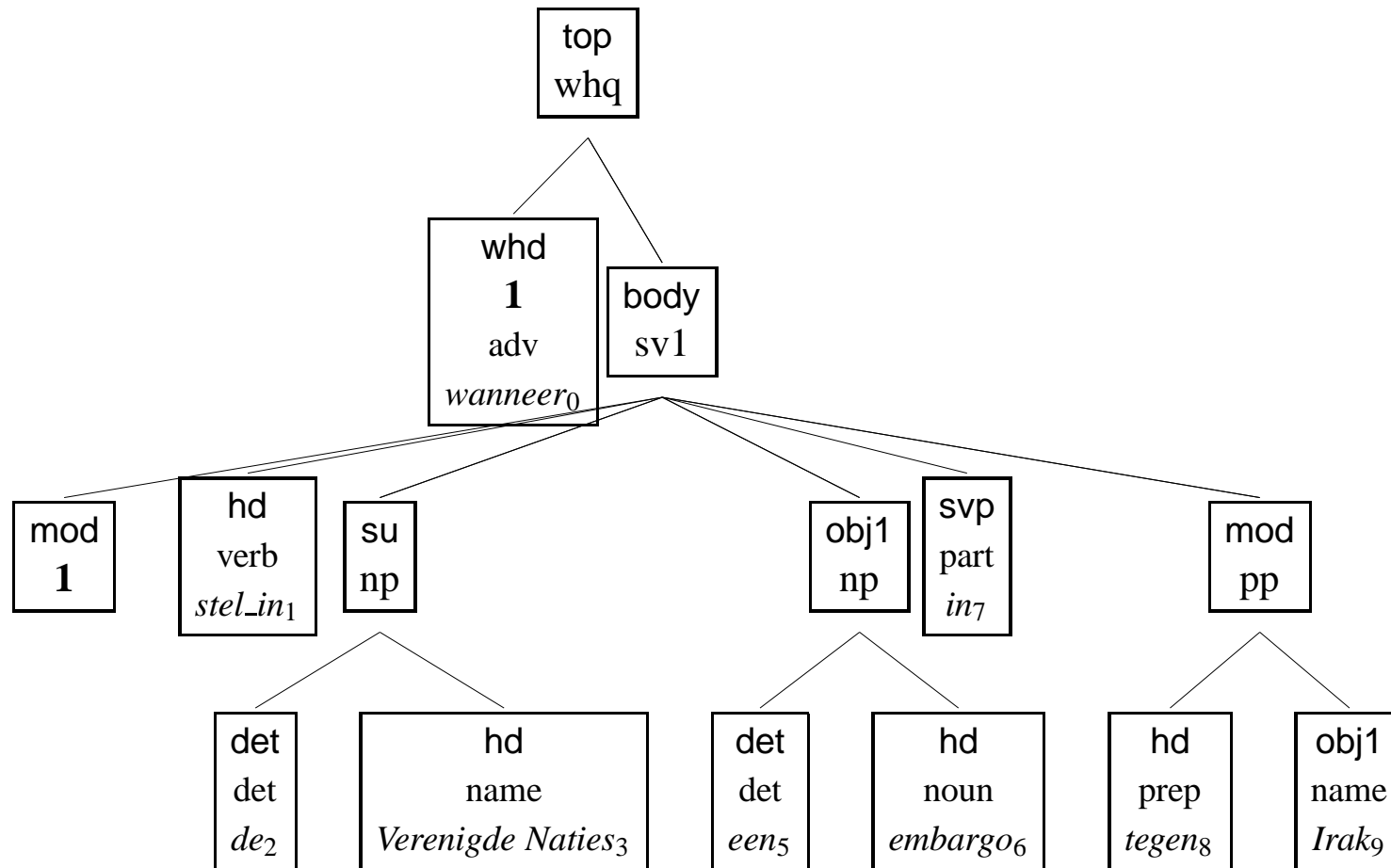
weighted: keywords can be weighted using a *boost factor*

required: keywords can be marked as required

proximity: a window can be defined for each set of (*restricted*) *token-layer* keywords

Query formulation - an example

Wanneer stelde de Verenigde Naties een embargo in tegen Irak?



Wanneer stelde de Verenigde Naties een embargo in tegen Irak?

● example query:

- (1) all plain *text* tokens
- (2) *RootHead* bigrams where the root is in an object relation
- (3) *RootRel* keywords for all nouns.
- (4) plain text names in a window of 50 tokens
- (5) NE-label corresponding to the question type

```
text:(stelde Verenigde Naties embargo Irak)
```

```
RootHead:(Irak/tegen embargo/stel_in)
```

```
RootRel:(embargo/obj1)
```

```
text:("Verenigde Naties Irak"~50)
```

```
neTypes:(YEAR)
```

Wanneer stelde de Verenigde Naties een embargo in tegen Irak?

- overlapping sub-queries:

- (1) all plain *text* tokens

- ...

- (6) plain text keywords in an object relation (boost factor 3)

- (7) plain text keywords labeled as names marked as required

```
text:(stelde +Verenigde +Naties embargo^3 Irak^3)
```

```
RootHead:(Irak/tegen embargo/stel_in)
```

```
RootRel:(embargo/obj1)
```

```
text:("Verenigde Naties Irak"~50)
```

```
neTypes:(YEAR)
```

Query optimisation

- How do we find the optimal way of combining keyword types? (there are > 200 possible types)
- How do we find the optimal boost factors for weighted keywords?
- How do we find the optimal window for proximity queries?

IR is a black box and we don't know exactly how parameter changes influence the retrieval performance.

Query optimisation

training: annotated CLEF questions

- 420 questions
- 360 of them with altogether 533 answers (doc-IDs and answer strings)

evaluation: disjoint set of CLEF questions

- 150 questions
- 126 of them with altogether 204 answers

Query optimisation

- training on CLEF2003 data using a simplified genetic algorithm (“trial&error beam search”):
 1. run initial queries (one keyword type per IR run) with default weights and window settings
 2. combine parameters of 2 of the N best IR runs (“**selection**” and “**crossover**”)
 3. change some settings at random from time to time (“**mutation**”)
 4. run the CLEF 2003 queries using the new settings and evaluate (determine “**fitness**”)
 5. continue with 2 until bored

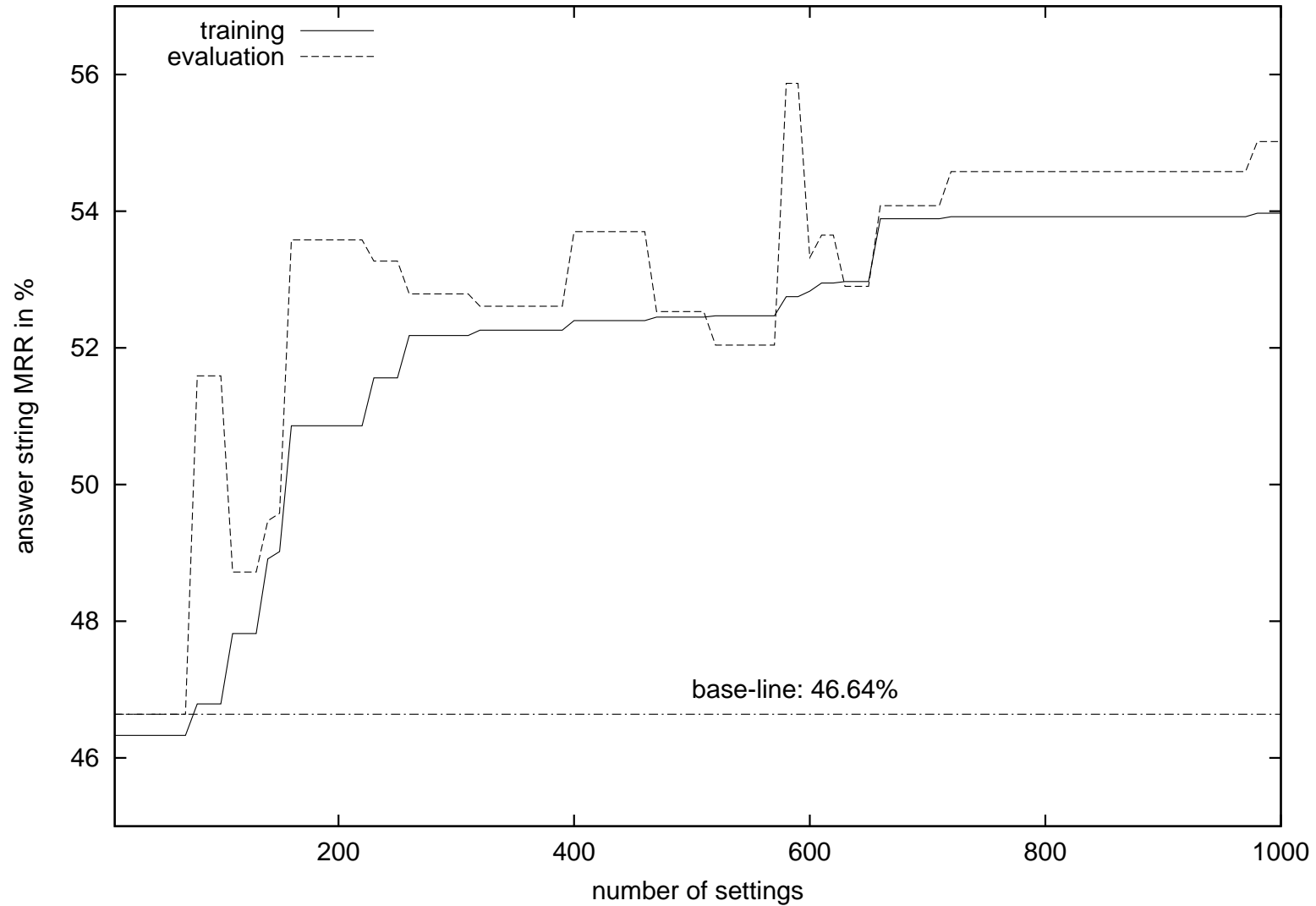
Query optimisation (results)

baseline = best performing single layer (text)

# settings	training (in %)	evaluation (in %)
baseline	46.33	46.64
50	45.71	48.79
100	46.79	51.59
250	51.56	53.27
500	52.45	52.53
750	53.92	54.58
1000	53.97	55.02

Very promising!

Query optimisation (results)



Multi-layer queries with Joost

running Joost with optimised multi-layer queries:

40 paragraphs	baseline (text)	optimised (1000)
training (strict)	0.357	0.363
(lenient)	0.459	0.468
evaluation (strict)	0.332	0.332
(lenient)	0.494	0.481

Quite a disappointment!

Multi-layer queries with Joost

evaluation data	baseline (text)	optimised (1000)
# paragraphs	strict MRR	strict MRR
top 40	0.332	0.332
top 20	0.299	0.320
top 10	0.297	0.308
top 5	0.295	0.315
top 4	0.290	0.323
top 3	0.276	0.320
top 2	0.257	0.303
top 1	0.199	0.302

Discussion & conclusions

- optimised queries with linguistic features improve precision
- ... but Joost's re-ranking cancels out the improved IR ranking
- ... however better IR reduces search space for Joost

→ work on less paragraphs (efficiency)

→ rely more on IR ranking (precision)

- more training data would be good
- other/more features?

THE END

Future work

- bagging & boosting in training (query optimisation)
- query expansion (similarity sets, relevance feedback, Wikipedia?)
- better text segmentation into proper paragraphs
- co-reference resolution in IR (?)
- any more ideas?

Query optimisation

small beam (population) size: 25

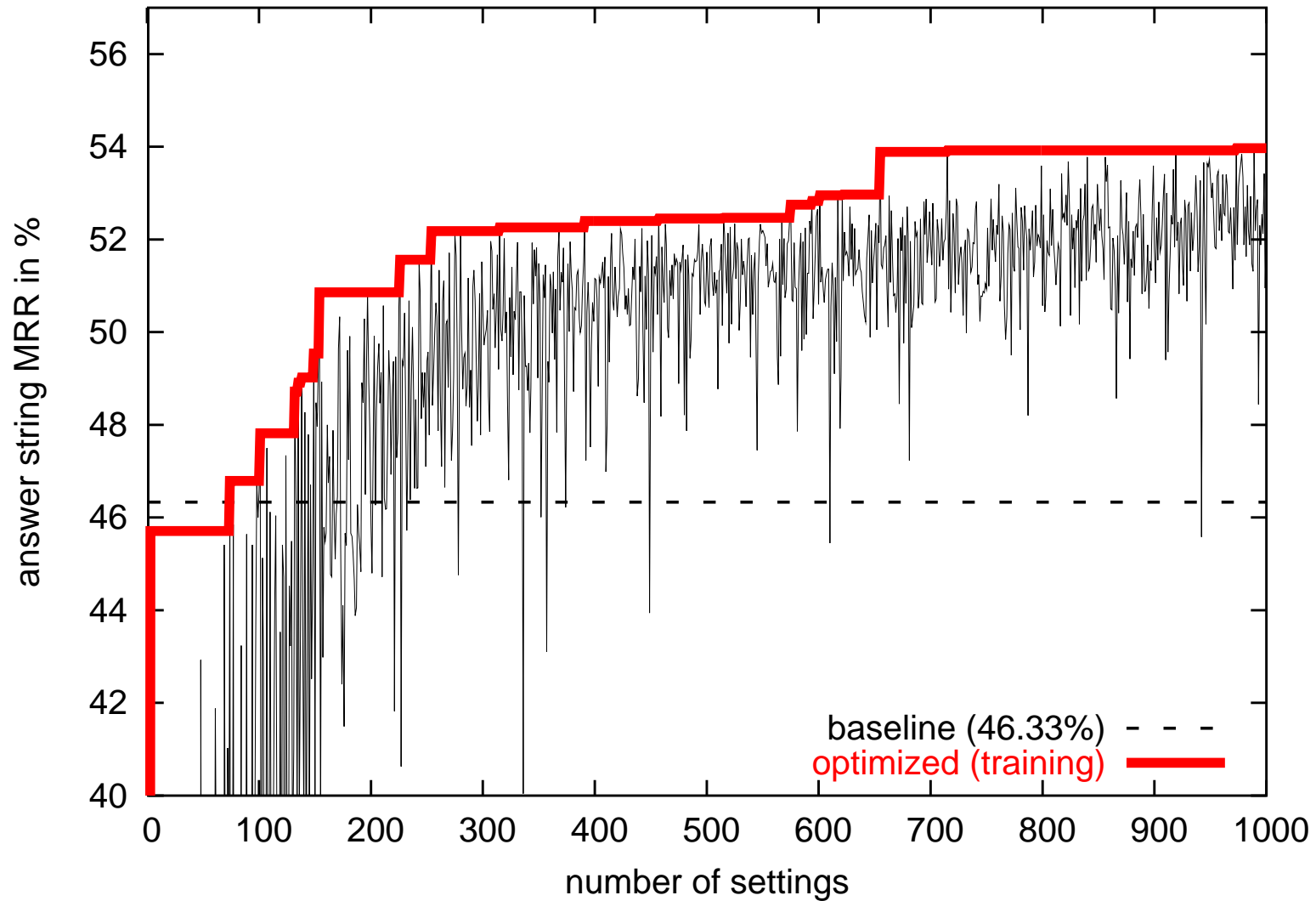
fitness scores: answerMRR (20 paragraphs)

selection: top 25 (always immediately after getting a new fitness score)

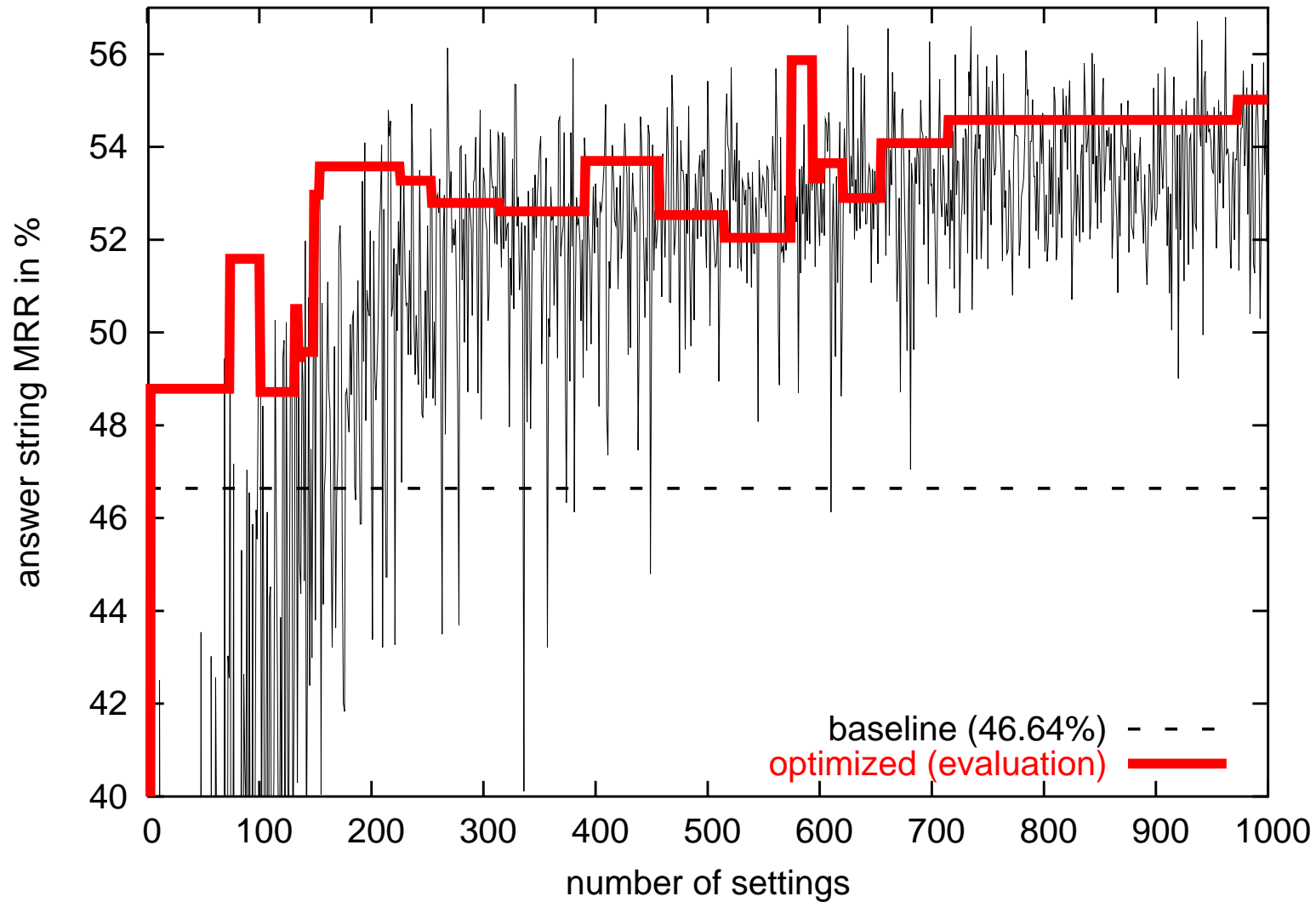
crossover: random selection of parents, arithmetic mean of weights/window sizes, max 25 new settings (children) simultaneously

mutation: add a keyword type (prob=0.2), remove a keyword type (prob=0.1), increase/decrease weights/window sizes (prob=0.2), set required marker (prob=0.01)

Query optimisation (training)



Query optimisation (evaluation)



Query parameters after 1000 settings

layer + restriction	weight/required	layer + restriction	window
text	8.81	text	15
root	+	text-mod	20
root-obj1	+	text_name-mod	15
root_noun-app	4.53	root	20
RootPos	+	root_name	15
RootPos_noun-su	0.95	root_noun-su	23
RootRel	+		
RootRel_name	+	RootRel_noun-obj1	20
RootHead_name	1.26	RootHead-su	23
RootHead_name-obj1	1	RootHead_name	22
RootRelHead	1.14	RootRelHead_noun-mod	17