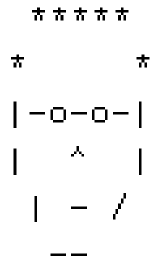
Exploiting the Web for Text and Language Reuse Applications

Benno Stein + Webis Group

Bauhaus-Universität Weimar
www.webis.de

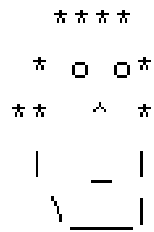
Webis Group

Webis Group



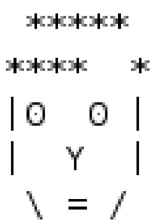
Benno Stein

Dennis Hoppe

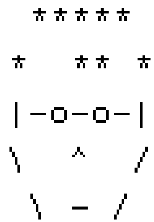


Maik Anderka

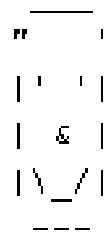
Martin Potthast



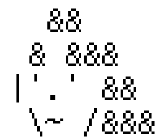
Matthias Hagen



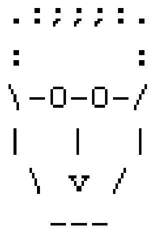
Nedim Lipka



Tim Gollub

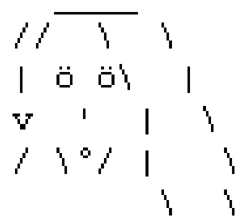


Tsvetomira Palakarska



Steven Burrows

Martin Trenkmann



Marie-Theresa Hansens

Nadin Glaser

Webis Group

Benno Stein

Dennis Hoppe

Maik Anderka

Martin Potthast

Matthias Hagen

```
  *****
 *  ** *
 |-o-o-|
 \  ^  /
  \ - /
```

Nedim Lipka

Tim Gollub

Tsvetomira Palakarska

Steven Burrows

Martin Trenkmann

Marie-Theresa Hansens

Nadin Glaser

Webis Group

```
    *****
  *       *
 |-o-o-|
 |   ^   |
  |  -  /
    --
```

Benno Stein

```
   """"""
 "  -00-"
 |     }  |
  \    -  /
     ---
```

Dennis Hoppe

```
    ****
  *  o  o*
 **    ^   *
  |    _   |
   \_____|
```

Maik Anderka

Martin Potthast

Matthias Hagen

```
    *****
  *    **   *
 |-o-o-|
 \    ^    /
   \  -  /
```

Nedim Lipka

```
    ____
  "       '
  | '   ' |
  |   &   |
  | \_/ |
    ---
```

Tim Gollub

```
    &&
   & &&&
 | ' . '  &&
  \~  /&&&
```

Tsvetomira Palakarska

```
 .:;;;:.
 :       :
 \-0-0-/
 |   |   |
  \  v  /
    ---
```

Steven Burrows

Martin Trenkmann

```
      _____
  //     \     \
  |  ö  ö\     |
 v    '     |    \
 /  \ °/   |      \
             \      \
```

Marie-Theresa Hansens

```
     .       .
   /  \_  ___
  /     a    a
 |          ^
 |  |     ___.
  \  \___      /
  /     |
```

Nadin Glaser

Talk

- The Netspeak Word Search Engine
- Query Segmentation
- Candidate Retrieval at PAN'12

Quicklinks

- Netspeak [video]
- ChatNoir
- PicaPica [video]

Webis Group

- Projects [local]
- Teaching [local]
- People [local]

The Netspeak Word Search Engine

The Netspeak Word Search Engine

Introduction

- Writing is not so much about what to write, but how to write.
- Finding the right words is essential to ease understanding.
- Searching for words is not well supported.

The Netspeak Word Search Engine

Introduction

- Writing is not so much about what to write, but how to write.
- Finding the right words is essential to ease understanding.
- Searching for words is not well supported.

We got for . . .

- spelling problems → spell checkers
- grammar mistakes → grammar checkers
- translation questions → dictionaries, machine translation tools
- word choice ambiguities → thesauri
- writing style analysis → reading and writing measures

The Netspeak Word Search Engine

Introduction

- Writing is not so much about what to write, but how to write.
- Finding the right words is essential to ease understanding.
- Searching for words is not well supported.

We got for . . .

- spelling problems → spell checkers
- grammar mistakes → grammar checkers
- translation questions → dictionaries, machine translation tools
- word choice ambiguities → thesauri
- writing style analysis → reading and writing measures

With Netspeak we address:

- finding the most common word (correctness vs. commonness)
- finding appropriate words in context

Netspeak Common Language Search

looks good ? me

Frequency		Phrase	Examples
56,925	83.6 %	looks good to me	⊞
10,047	14.8 %	looks good on me	⊞
1,123	1.6 %	looks good for me	⊞
68,095	100.0 %		0.048 seconds

Home Examples Developer Terms Contact Us

[www.netspeak.org]

The Netspeak Word Search Engine

Corpus Statistics

- Netspeak is based on the Google n-gram corpus "Web 1T 5-gram Version 1" from 2006:

Subset	n-gram number	Space	after postprocessing
1-gram	13 588 391	177.0 MB	3.75 %
2-gram	314 843 401	5.0 GB	43.26 %
3-gram	977 069 902	19.0 GB	48.65 %
4-gram	1 313 818 354	30.5 GB	49.54 %
5-gram	1 176 470 663	32.1 GB	47.16 %
Σ	3 354 253 200	77.9 GB	54.20 %

- Postprocessing includes:
  - all n-grams: conversion to lower case
  - 1-grams: deletion of words with frequency below 3 000
    deletion of words with certain special characters (blacklist)
    $\rightarrow$ about 1.5 million 1-grams remain
  - 2/3/4/5-grams: filtering with respect to the 1.5 million 1-grams

- See Netspeak load statistics online.

The Netspeak Word Search Engine

Related Research

Wildcard search	[Cafarella and Etzioni, WWW 2005]
	[Resnik and Elkiss, ACL 2005]
	[Rafiei and Li, CIKM 2009]
	[Tsang and Chawla, CIKM 2011]
	[Vaele, ACL 2011]
n-Gram indexing	[Lin et al., LREC 2010]
	[Hawker et al., ALT 2007]
	[Carlson, Carnegie Mellon 2008]
	[Brants, EMNLP/CONLL 2007]
	[Guthrie and Hepple, EMNLP 2010]
Error correction	[Leacock et al., Morgan & Claypool 2010]
	[Brockett et al., ACL 2006]
Digital humanities	[Michel et al., Science 2010]

The Netspeak Word Search Engine

Problem Statement

Given: A set D of n-grams, $n \leq 5$, with frequencies $f : D \rightarrow \mathbf{N}$.
A query q as a sequence of words and wildcards.

query	=	$\{$ word $\mid$ wildcard $\}_1^5$
wildcard	=	"?" $\mid$ "$*$" $\mid$ synonyms $\mid$ multiset $\mid$ optionset
synonyms	=	"#" word
multiset	=	"$\{$" word $\{$ word $\}$ "$\}$"
optionset	=	"[" word $\{$ word $\}$ "]"

Task: Retrieve all n-grams in D that match q.

The Netspeak Word Search Engine

Problem Statement

Given: A set D of n-grams, $n \leq 5$, with frequencies $f : D \to \mathbf{N}$.
A query q as a sequence of words and wildcards.

query	=	$\{$ word | wildcard $\}_1^5$
wildcard	=	"?" | "$*$" | synonyms | multiset | optionset
synonyms	=	"#" word
multiset	=	"{" word $\{$ word $\}$ "}"
optionset	=	"[" word $\{$ word $\}$ "]"

Task: Retrieve all n-grams in D that match q.

Straightforward solution:

1. Construct an inverted index $\mu : V \to \mathcal{P}(D)$, where V is D's vocabulary.
2. Retrieve the n-grams $R = \bigcap_{w \in q} \mu(w)$ that contain all of q's words $w \in V$.
3. Compile a pattern matcher from q and filter R.

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ `hello kitty ?`

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$$q = \texttt{hello kitty ?} \rightarrow \mu(\texttt{hello})$$
$$\rightarrow \mu(\texttt{kitty})$$

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ `hello kitty ?` $\rightarrow \boxed{\mu(\texttt{hello})} \rightarrow$
$\rightarrow \mu(\texttt{kitty})$

hello	33 000 000
hello all	800 000
hello to all	140 000
say hello	1 000 000
posted by hello	339 000
say hello to	374 000

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ `hello kitty ?` $\rightarrow \boxed{\mu(\texttt{hello})} \rightarrow$
$\rightarrow \mu(\texttt{kitty})$

~~hello~~	~~33 000 000~~
~~hello all~~	~~800 000~~
hello to all	140 000
~~say hello~~	~~1 000 000~~
posted by hello	339 000
say hello to	374 000

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ hello kitty ? $\rightarrow$	$\boxed{\mu(\texttt{hello})}$ $\rightarrow$	~~hello~~	~~33 000 000~~	1-gram index
$\rightarrow$	$\mu(\texttt{kitty})$	~~hello all~~	~~800 000~~	2-gram index
		hello to all	140 000	3-gram index
		~~say hello~~	~~1 000 000~~	4-gram index
		posted by hello	339 000	5-gram index
		say hello to	374 000	

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q = \texttt{hello kitty ?} \rightarrow \boxed{\mu(\texttt{hello})}$	$\rightarrow$ ~~hello~~	~~33 000 000~~			1-gram index
$\rightarrow \mu(\texttt{kitty})$	~~hello all~~	~~800 000~~			2-gram index
	hello to all	140 000	$\leftarrow \mu_3(\texttt{hello})$	$\leftarrow$	$\boxed{\text{3-gram index}}$
	~~say hello~~	~~1 000 000~~			4-gram index
	posted by hello	339 000	$\leftarrow$		5-gram index
	say hello to	374 000	$\leftarrow$		

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ `hello * kitty *`

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

```
q = hello * kitty *  →  hello kitty
                        hello ? kitty
                        hello kitty ?
                        hello ? ? kitty
                        hello ? kitty ?
                        hello kitty ? ?
                        hello ? ? ? kitty
                        hello ? ? kitty ?
                        hello ? kitty ? ?
                        hello kitty ? ? ?
```

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ `hello * kitty *` $\rightarrow$

```
hello kitty
hello [? kitty]     → ? ? kitty
hello [kitty ?]     → ? kitty ?
hello ? ? kitty
hello ? kitty ?
hello kitty ? ?
hello ? ? ? kitty
hello ? ? kitty ?
hello ? kitty ? ?
hello kitty ? ? ?
```

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys

$q =$ `hello * kitty *` $\rightarrow$ `hello kitty`

`hello` $\boxed{\texttt{? kitty}}$ $\rightarrow$ `? ? kitty` $\rightarrow$ $\mu_{3P}(\texttt{kitty}, 3, 2)$

`hello` $\boxed{\texttt{kitty ?}}$ $\rightarrow$ `? kitty ?` $\rightarrow$ $\mu_{3P}(\texttt{kitty}, 3, 1)$

`hello ? ? kitty`

`hello ? kitty ?`

`hello kitty ? ?`

`hello ? ? ? kitty`

`hello ? ? kitty ?`

`hello ? kitty ? ?`

`hello kitty ? ? ?`

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys
- Exploit task characteristics: trade retrieval recall for retrieval time

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys
- Exploit task characteristics: trade retrieval recall for retrieval time

$q =$ `hello kitty ?`

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys
- Exploit task characteristics: trade retrieval recall for retrieval time

$$q = \texttt{hello kitty ?} \;\rightarrow\; \mu_{3P}(\texttt{hello}, 3, 0)$$
$$\rightarrow\; \mu_{3P}(\texttt{kitty}, 3, 1))$$

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys
- Exploit task characteristics: trade retrieval recall for retrieval time

q = `hello kitty ?` $\rightarrow$ $\boxed{\mu_{3P}(\texttt{hello}, 3, 0)}$ $\rightarrow$
$\rightarrow$ $\mu_{3P}(\texttt{kitty}, 3, 1))$

`hello, my`	500 000
`hello and welcome`	261 000
`hello world!`	175 000
`hello ...`	...
`hello my friend`	13 000
`hello kitty princess`	8 400
`hello is there`	4 600
`hello and how`	2 600
`hello is any`	320

The Netspeak Word Search Engine

Index Construction

Considerations and implications:

- Exploit closed retrieval setting (D is constant): perfect hashing
- Exploit small n-gram lengths: compile filtering effort into index
  - multiple indexes for fixed query lengths
  - uniform query representation by enfolding
  - positional subqueries as index keys
- Exploit task characteristics: trade retrieval recall for retrieval time

q = `hello kitty ?` $\rightarrow$ $\boxed{\mu_{3P}(\texttt{hello}, 3, 0)}$ $\rightarrow$
$\rightarrow$ $\mu_{3P}(\texttt{kitty}, 3, 1))$

`hello, my`	500 000
`hello and welcome`	261 000
`hello world!`	175 000
`hello ...`	...
~~`hello my friend`~~	~~13 000~~
~~`hello kitty princess`~~	~~8 400~~
~~`hello is there`~~	~~4 600~~
~~`hello and how`~~	~~2 600~~
~~`hello is any`~~	~~320~~

The Netspeak Word Search Engine

Index Construction

Examples:

n-gram	Frequency	Identifier	Postlist entry	Index	Key
`hello`	33 000 000	1	(1, 33 000 000)	symbol table	`hello`
`world`	432 000 000	3	(3, 432 000 000)	symbol table	`world`
`hello world`	712 963	4	(4, 712 963)	symbol table	`hello world`
				2-gram	(`hello`, 2, 0)
				2-gram	(`world`, 2, 1)

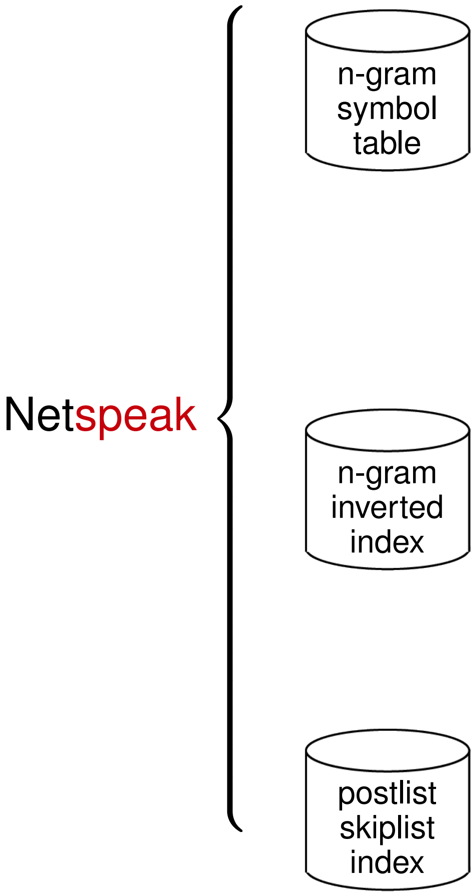
Part of the symbol table is used as 1-gram index at the same time.

Netspeak index statistics:

- 2 013 781 863 keys in n-gram symbol table
- 19 346 361 keys in 2/3/4/5-gram indexes
- 7 782 365 325 postlist entries
- 134 GB index size, 1.7 GB memory footprint

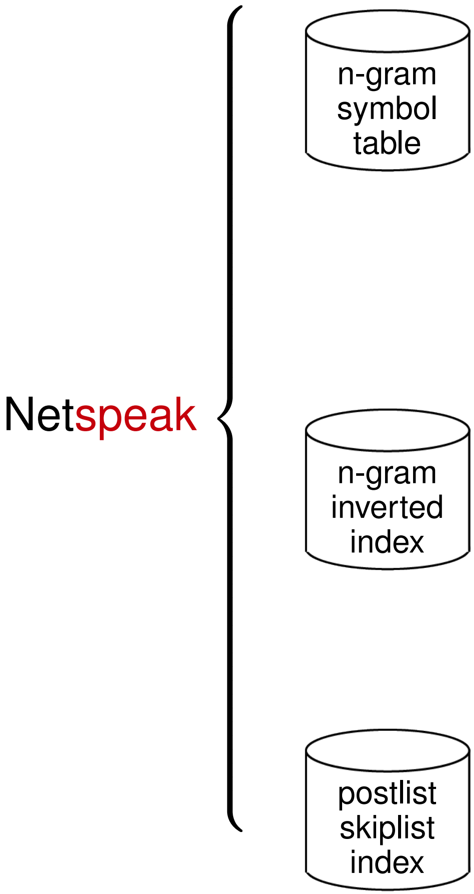
The Netspeak Word Search Engine

Index Construction



The Netspeak Word Search Engine

Index Construction



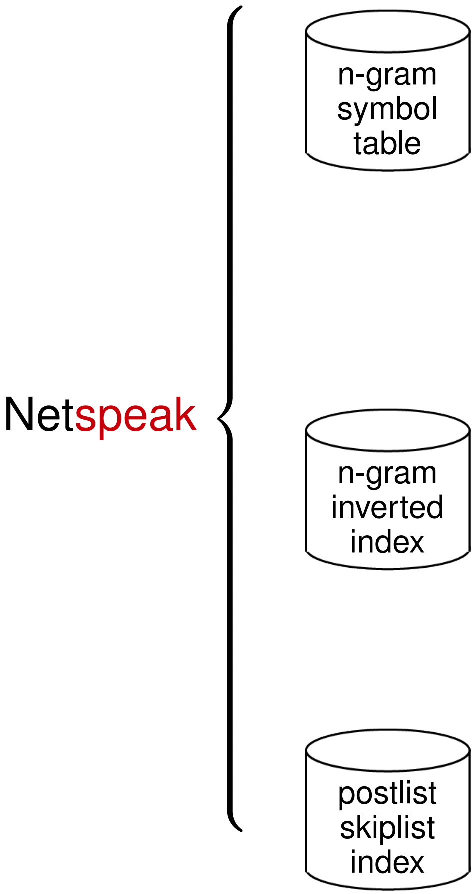
Bijective mapping between all n-grams and their postlist entries.
I.e., frequency information is symbol table payload.

Example: `hello world` $\mapsto$ (4, 712 963) // n-gram $\mapsto$ (id, frequency)
(4, 712 963) $\mapsto$ `hello world` // (id, frequency) $\mapsto$ n-gram

Rationale: Follow design of inverted indexes.
Lookup table for non-wild-card queries.
Database with underestimations for skip heuristics.

The Netspeak Word Search Engine

Index Construction



Bijective mapping between all n-grams and their postlist entries.
I.e., frequency information is symbol table payload.

Example: `hello world` $\mapsto$ (4, 712 963) // n-gram $\mapsto$ (id, frequency)
(4, 712 963) $\mapsto$ `hello world` // (id, frequency) $\mapsto$ n-gram

Rationale: Follow design of inverted indexes.
Lookup table for non-wild-card queries.
Database with underestimations for skip heuristics.

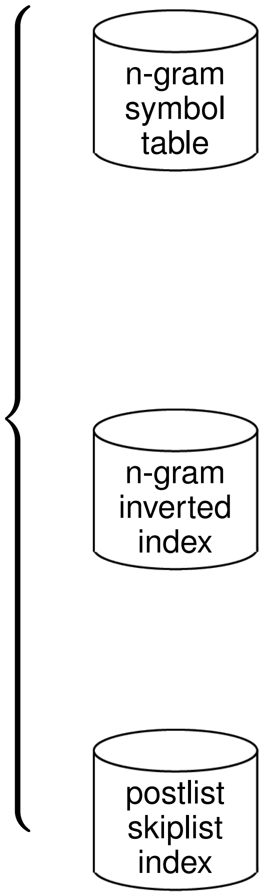
Inverted index where n-grams are in the role of documents.
A key encodes also n-gram length and word position.
A postlist is a sorted list of (id, frequency)-tuples.

Example: (`hello`, 2, 0) $\mapsto$ ((5, 1 469 134), (4, 712 963))
(`world`, 2, 1) $\mapsto$ ((4, 712 963))

The Netspeak Word Search Engine

Index Construction

Netspeak



Bijective mapping between all n-grams and their postlist entries.
I.e., frequency information is symbol table payload.

Example: `hello world` $\mapsto$ (4, 712 963) // n-gram $\mapsto$ (id, frequency)
(4, 712 963) $\mapsto$ `hello world` // (id, frequency) $\mapsto$ n-gram

Rationale: Follow design of inverted indexes.
Lookup table for non-wild-card queries.
Database with underestimations for skip heuristics.

Inverted index where n-grams are in the role of documents.
A key encodes also n-gram length and word position.
A postlist is a sorted list of (id, frequency)-tuples.

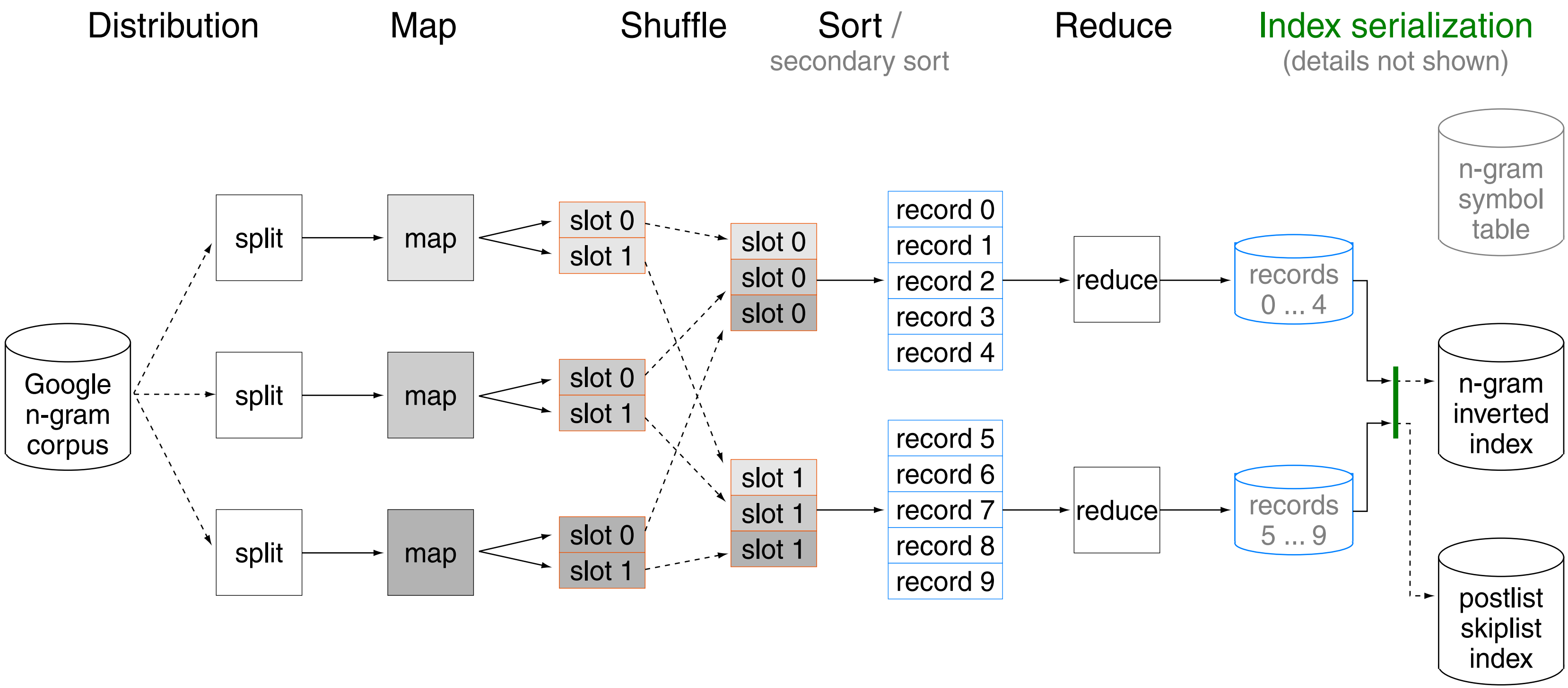
Example: (`hello`, 2, 0) $\mapsto$ ((5, 1 469 134), (4, 712 963))
(`world`, 2, 1) $\mapsto$ ((4, 712 963))

Meta index that specifies postlist entry points according to the n-gram frequency distribution.

Rationale: Efficient search for set operations on sorted postlists.
Implementation of frequency range queries.

The Netspeak Word Search Engine

Construction Pipeline



Input records

(line number, line)

(1, `hello world` 712963 4)
(2, `hello kitty` 1468134 5)

Intermediate records

((word, len, pos), (freq, id))

((`hello`, 2, 0), (712963, 4))
((`world`, 2, 1), (712963, 4))
((`hello`, 2, 0), (1468134, 5))
((`kitty`, 2, 1), (1468134, 5))

Sorted records

((word, len, pos), ((freq, id) ... (freq, id)))

((`hello`, 2, 0), ((1468134, 5),(712963, 4)))
((`world`, 2, 1), ((712963, 4)))
((`kitty`, 2, 1), ((1468134, 5)))

Output records

(reduced sorted records)

- - - → Cluster-wide data copy
——→ Local data copy

The Netspeak Word Search Engine

Selected Index Performance Issues

- n-gram symbol table (external)
  - n-gram $\mapsto$ (id, frequency)
    minimal perfect hash functions via BDZ algorithm [Belazzougui/Botelho/Dietzfelbinger 2009]
  - (id, frequency) $\mapsto$ n-gram
    four external arrays encoding the 2/3/4/5-grams via their 1-gram ids

- n-gram inverted index (external)
  - tailored indexing of 2/3/4/5-grams that enable positional subquery lookup
  - header tables that map positional subqueries to postlists on the harddisk

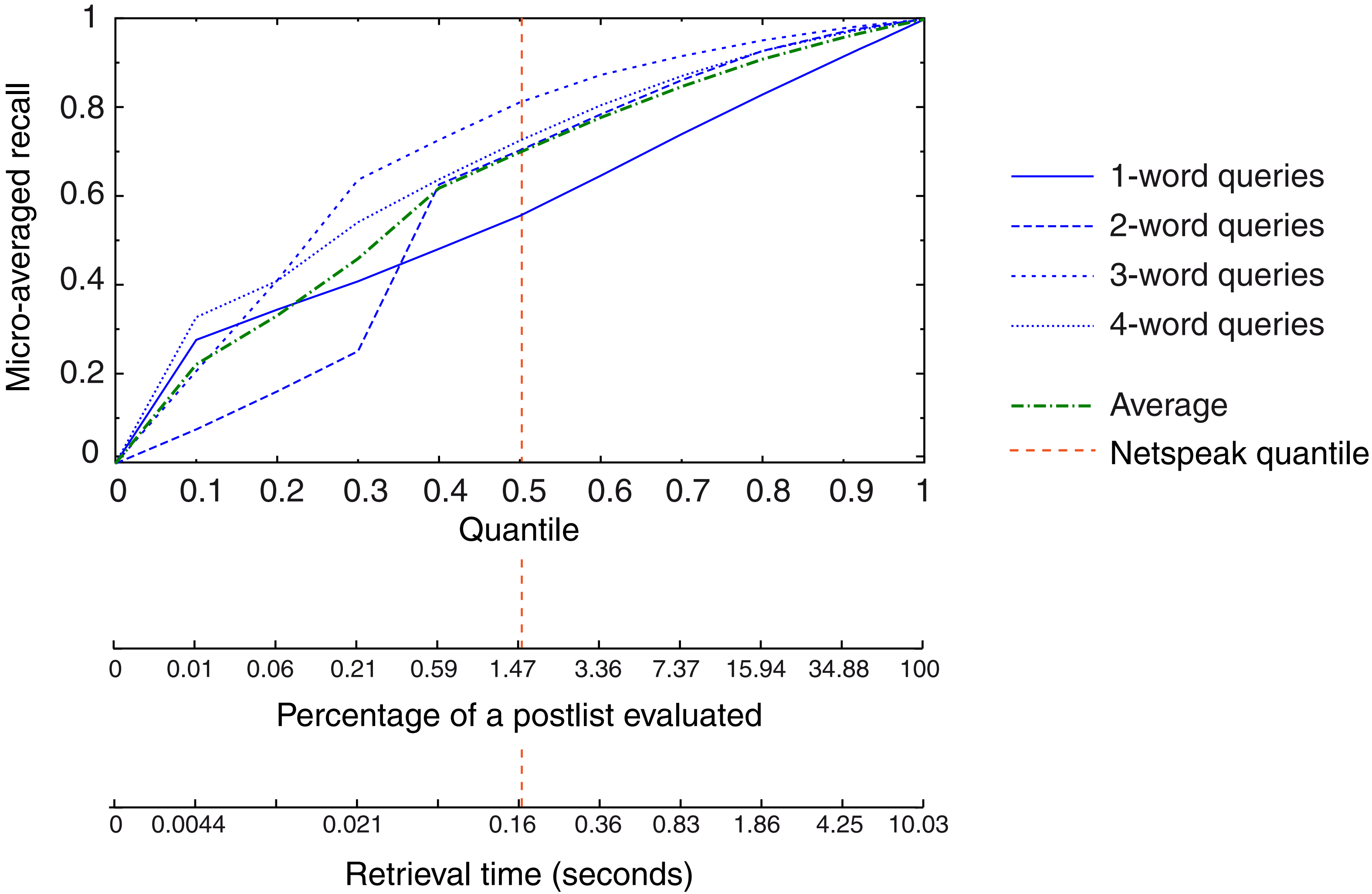
- postlist skiplist index (external)
  - applies to postlists that fit not into memory (> 10^5 entries)
  - skip density models frequency distribution and enables task-specific pruning strategies
  - ➜ heuristic retrieval that considers the number of matches or postlist entries
  - ➜ heuristic retrieval that considers the word class. Pruning strategy for phrase ranking:

immediate response	non-stopword postlist	0.80 quantile
	stopword postlist	0.30 quantile
near 1-recall response	non-stopword postlist	0.95 quantile
	stopword postlist	0.50 quantile

- result caching

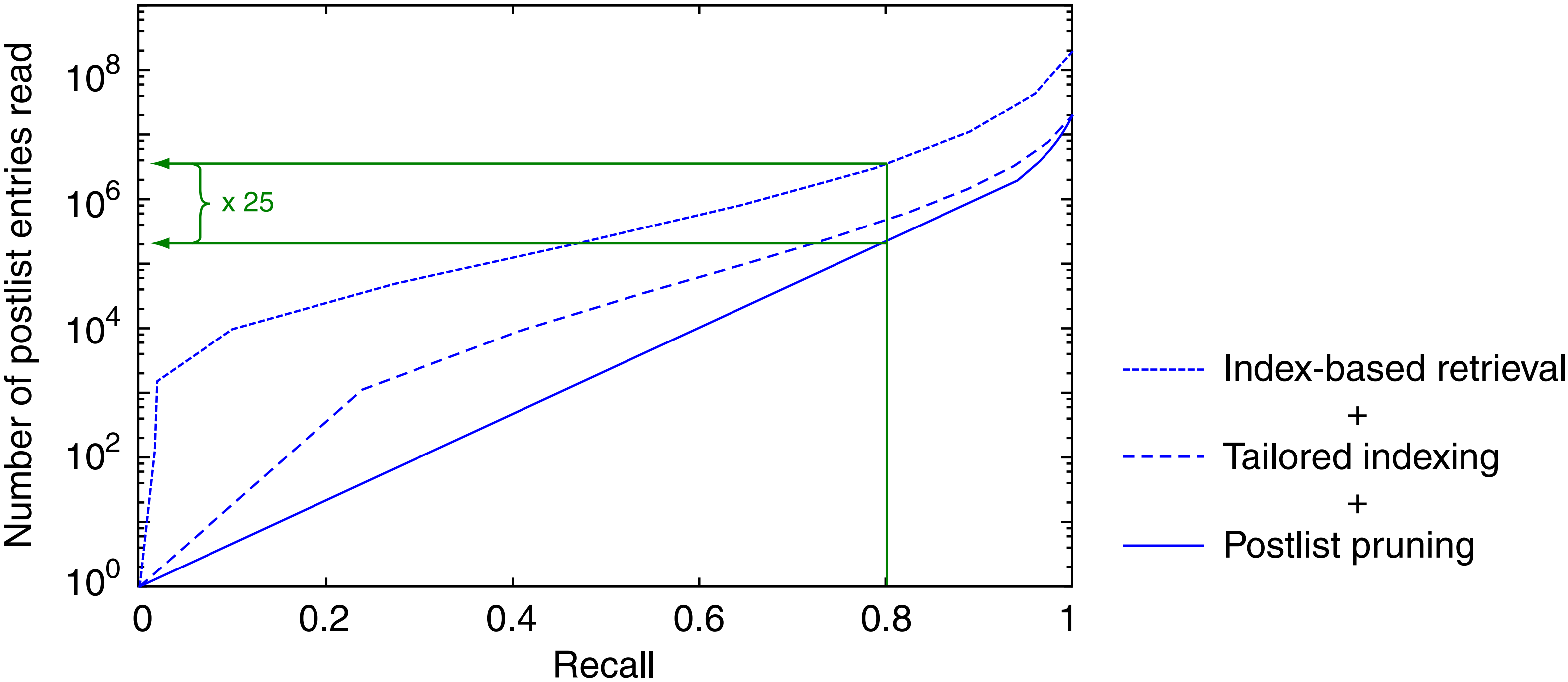
The Netspeak Word Search Engine

Retrieval Performance—Data Perspective: Completeness → Recall



The Netspeak Word Search Engine

Retrieval Performance—Technology Perspective: Effort → Recall



- reduction of retrieval effort by factor 25 due to indexing technology

The Netspeak Word Search Engine

Netspeak in a Nutshell

Given: A set D of n-grams, $n \leq 5$, with frequencies $f : D \rightarrow \mathbf{N}$.
A query q as a sequence of words and wildcards.

Task: Retrieve all n-grams in D that match q.

Solution:

- Construct an inverted index $\mu : V \times \underbrace{\{1, \ldots, 5\}}_{\text{n-gram length}} \times \underbrace{\{1, \ldots, 5\}}_{\text{word position}} \rightarrow \mathcal{P}(D)$
- Sort $\mu(w, i, j)$ in descending order of f, where $w \in V$ and $i, j \in \{1, \ldots, 5\}$.
- Let R_q denote the true retrieval result for q.
- Enfold q into $\{q_1, \ldots, q_m\}$ such that $\bigcup_{i=1}^{m} R_{q_i} = R_q$, and each q_i matches only n-grams with a fixed length. Process sub-queries in parallel.
- Retrieve all n-grams $R_{q_i} = \bigcap_{w \in q_i} \mu(w, |q_i|, q_{i|w})$, with $q_{i|w}$ denoting w's position in q_i.
- Process $\mu(w, i, j)$ starting at rightmost entry k, where $f(\mu(w, i, j)_k) \leq \min_{d \in \{q_1, \ldots, q_m\}}(f(d))$.
- Stop processing $\mu(w, i, j)$ at entry

$$\begin{cases} |\mu(w, i, j)| & \text{if the postlist is smaller than a page, or} \\ l_1 & \text{if a pre-specified amount of results have been retrieved, or} \\ l_2 & \text{if } \sum_{i'=0}^{l_2} f(\mu(w, i, j)_{i'}) \text{ covers } \kappa\% \text{ of the frequency distribution.} \end{cases}$$

The Netspeak Word Search Engine

Netspeak Summary

Netspeak's targeted users want to improve their writing:

- scientists, authors, scholars, journalists, bloggers

I.e., at the beginning we were thinking of following use cases:

- scientists who speak English as a second language
- scientists who ask what is commonly written in their research field
- a Netspeak that is tailored to the genre of scientific writing
- support for corpus-linguistic research

The Netspeak Word Search Engine

Netspeak Summary

Netspeak's targeted users want to improve their writing:

- scientists, authors, scholars, journalists, bloggers

I.e., at the beginning we were thinking of following use cases:

- scientists who speak English as a second language
- scientists who ask what is commonly written in their research field
- a Netspeak that is tailored to the genre of scientific writing
- support for corpus-linguistic research

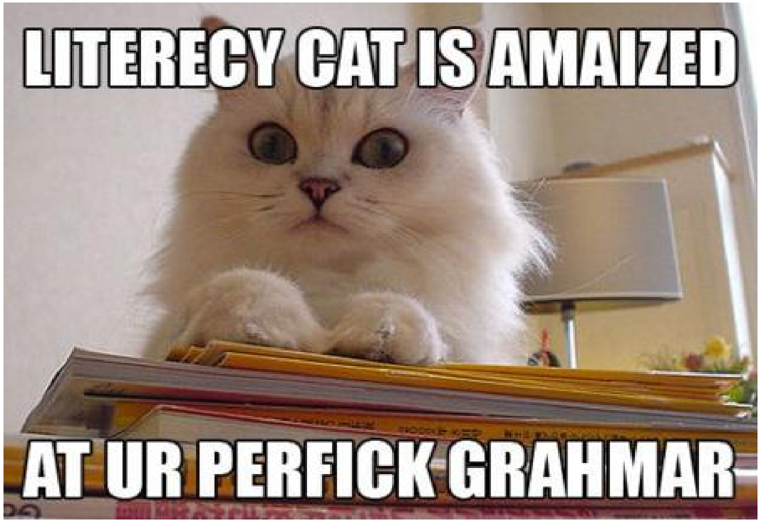
Meanwhile we are also thinking of the following (and more):

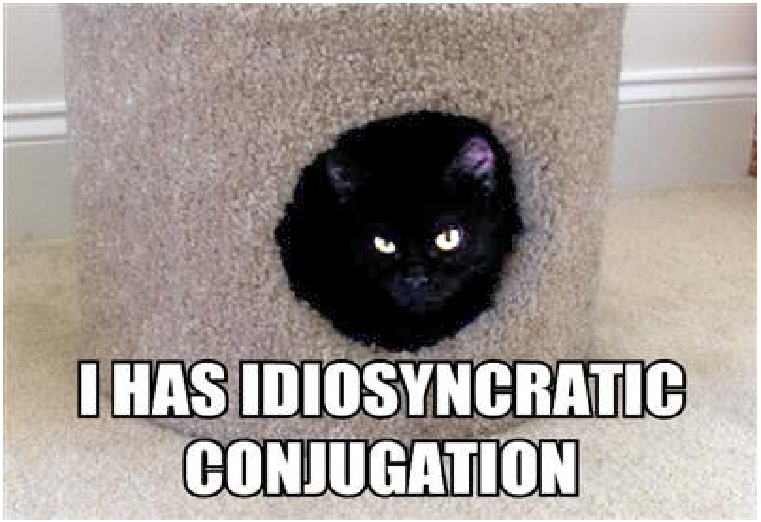
- query segmentation and query cover
- paraphrase generation and evaluation
- POS search, morphological search, partial word search

Research on subquery compilation: key space size versus filtering effort

The Netspeak Word Search Engine

What Our Users Say ;–)





Query Segmentation

Query Segmentation

What is the User Searching?

```
new york times square dance
```

Query Segmentation

What is the User Searching?

```
new york times square dance
```

All search engines face the same problem.

Query Segmentation

What is the User Searching?

`new york    times square    dance`

Image source: [http://www.theepochtimes.com/n2/images/stories/large/2009/08/06/Bollywood1.jpg]

Query Segmentation

What is the User Searching?

```
new york times     square dance
```

"All the News That's Fit to Print"

The New York Times

Late Edition
Today, a shower, yielding to sun, high 64. Tonight, chilly, low 45. Tomorrow, sunny and cool, high 64. Yesterday's high, 70, low, 54. Weather map and details, Page 24.

VOL. CLVIII . No. 54,678 ©2009 The New York Times NEW YORK, SUNDAY, MAY 17, 2009 $5 beyond the greater New York metropolitan area. $4.00

Square Dance

19 U.S. States Have Designated It As Their Official State Dance

From a Theory To a Consensus On Emissions

Permits Gain Political Edge Over Taxation

By JOHN M. BRODER

WASHINGTON — As Congress weighs imposing a mandatory limit on climate-altering gases — an outcome still far from certain — it is likely to turn to a system that sets a government ceiling on total emissions and allows polluting industries to buy and sell permits to meet it.

That approach, known as cap and trade, has been embraced by President Obama, Democratic leaders in Congress, mainstream environmental groups and a growing number of business interests, including energy-consuming industries like autos, steel and aluminum.

But not long ago, many of today's supporters dismissed the idea of tradable emissions permits as an industry-inspired Republican scheme to avoid the real costs of cutting air pollution. The right answer, they said, was strict government regulation, state-of-the-art technology and a federal tax on every ton of harmful emissions.

How did cap and trade, hatched as an academic theory in obscure economic journals half a century ago, become the policy of choice in the debate over how to slow the heating of the planet? And how did it come to eclipse the idea of simply slapping a tax on energy consumption that befouls the public square or leaves the nation hostage to foreign oil

CONSERVATIVES MAP STRATEGIES ON COURT FIGHT

MEMOS OUTLINE ATTACKS

Hoping to Re-Energize G.O.P. by Opposing Obama's Choice

By CHARLIE SAVAGE

WASHINGTON — If President Obama nominates Judge Diane P. Wood to the Supreme Court, conservatives plan to attack her as an "outspoken" supporter of "abortion, including partial-birth abortion."

If he nominates Judge Sonia Sotomayor, they plan to accuse her of being "willing to expand constitutional rights beyond the text of the Constitution."

And if he nominates Kathleen M. Sullivan, a law professor at Stanford, they plan to denounce her as a "prominent supporter of homosexual marriage."

Preparing to oppose the confirmation of Mr. Obama's eventual choice to succeed Justice David H. Souter, who is retiring, conservative groups are working together to stockpile ammunition. Ten memorandums summarizing their research, obtained by The New York Times, provide a window onto how they hope to frame the coming debate.

The memorandums dissect possible nominees' records, noting statements the groups find objectionable on issues like abortion, same-sex marriage, the sen-

Image source: [http://blog.caseytempleton.com/wp-content/uploads/2009/05/090517_nytfrontpage1.jpg]
Image source: [http://upload.wikimedia.org/wikipedia/commons/0/03/Square_Dance_Group.jpg]

Query Segmentation

Segment Your Queries!

The benefits:

- improved precision
- potential disambiguation
- reformulation at segment level

The syntax:

- Quotes around segments: `"new york" "times square" dance`

Query Segmentation

Segment Your Queries!

The benefits:

- improved precision
- potential disambiguation
- reformulation at segment level

The syntax:

- Quotes around segments: `"new york" "times square" dance`

The reality:

- Most web searchers are not even aware of the quotes option.

Query Segmentation

Segment Your Queries!

The benefits:

- improved precision
- potential disambiguation
- reformulation at segment level

The syntax:

- Quotes around segments: `"new york" "times square" dance`

The reality:

- Most web searchers are not even aware of the quotes option.

The solution:

- Automatic *pre-retrieval* query segmentation. (response time is indispensable)

Query Segmentation

Segment Your Queries!

Given: A keyword query.

Task: Find the “best” segmentation.

Boundary conditions:

- we assume correct spelling
- we do not change keywords
- we do not change word order

Examples:

Query	`new york times square dance`	
Valid candidates	`"new york" "times square" dance`	(three segments)
	`"new york times" "square dance"`	(two segments each)
No candidate	`"new york" "dance times square"` (a Latin dance studio in NYC)	

Query Segmentation

Related Research

Mutual information	[Risvik et al., WWW 2003] [Jones et al., WWW 2006] [Huang et al., WWW 2010]
Supervised learning	[Bergsma and Wang, EMNLP-CoNLL 2007] [Bendersky et al., SIGIR 2009]
Unsupervised learning	[Tan and Peng, WWW 2008] [Zhang et al., ACL-IJCNLP 2009]
Retrieval feedback	[Brenes et al., CERI 2010] [Bendersky et al., CIKM 2010] [Bendersky et al., ACL 2011]
Query log	[Mishra et al., WWW 2011] [Li et al., SIGIR 2011]

Query Segmentation

Our first approach [Hagen et al., SIGIR 2010]

. . . follows a well-known principle.

KISS—keep it simple and stupid—and use the Web as a corpus.

Image source: [http://1.bp.blogspot.com/_UDZXrzYpS4k/THRrh8KPvVI/AAAAAAAAAoc/Be1HjlnRy1c/s400/lipstick-mirror.jpg]

Query Segmentation

Exploiting Web Phrase Frequencies [Google n-grams, Brants and Franz, LDC 2006]

Rationale:

(a) web phrases = reasonable segments

(b) more frequent = better segments

Query Segmentation

Exploiting Web Phrase Frequencies [Google n-grams, Brants and Franz, LDC 2006]

Rationale:

(a) web phrases = reasonable segments

(b) more frequent = better segments

Summing up raw frequencies won't yield a sensible ranking:

`"new york" times` = 165.4 million

`"new york times"` = 17.5 million

(short segments always win)

Query Segmentation

Exploiting Web Phrase Frequencies

[Google n-grams, Brants and Franz, LDC 2006]

Rationale:

(a) web phrases = reasonable segments

(b) more frequent = better segments

Summing up raw frequencies won't yield a sensible ranking:

`"new york" times` = 165.4 million

`"new york times"` = 17.5 million

(short segments always win)

From the data we derived the following segment frequency normalization:

$|s|^{|s|} \cdot \mathit{freq}(s)$ (s denotes a segment)

Query Segmentation

Exploiting Web Phrase Frequencies

Examples:

segment s	*freq*(s)
`new york`	165.4 million
`new york times`	17.5 million
`new york times square`	20 476
`new york times square dance`	0
`york times`	17.6 million
`york times square`	20 561
`york times square dance`	0
`times square`	1.3 million
`times square dance`	104
`square dance`	210 440

Query Segmentation

Exploiting Web Phrase Frequencies

Examples:

segment s	*freq*(s)	$\|s\|^{\|s\|} \cdot$ *freq*(s)
`new york`	165.4 million	661.6 million
`new york times`	17.5 million	472.5 million
`new york times square`	20 476	5.2 million
`new york times square dance`	0	0
`york times`	17.6 million	70.4 million
`york times square`	20 561	0.5 million
`york times square dance`	0	0
`times square`	1.3 million	5.2 million
`times square dance`	104	2 808
`square dance`	210 440	0.8 million

Query Segmentation

Exploiting Web Phrase Frequencies

The normalized segment frequencies sum up to the *score* of a query S:

$$\textit{score}(S) = \begin{cases} \displaystyle\sum_{s \in S,\ |s| \geq 2} |s|^{|s|} \cdot \textit{freq}(s) & \\ -1 & \text{if } |s| \geq 2 \wedge \textit{freq}(s) = 0 \text{ for some } s \in S \end{cases}$$

Ranking:

rank	segmentation S	*score*(S)
1	`"new york" "times square" dance`	666.8 million
2	`"new york" times "square dance"`	662.4 million
⋮	⋮	⋮
5	`"new york times" "square dance"`	473.3 million
⋮	⋮	⋮
13	`new york "times square dance"`	2 808
14	`new york times square dance`	0
15	`"new york times square dance"`	-1
16	`new "york times square dance"`	-1

Query Segmentation

Our second approach [Hagen et al., WWW 2011]

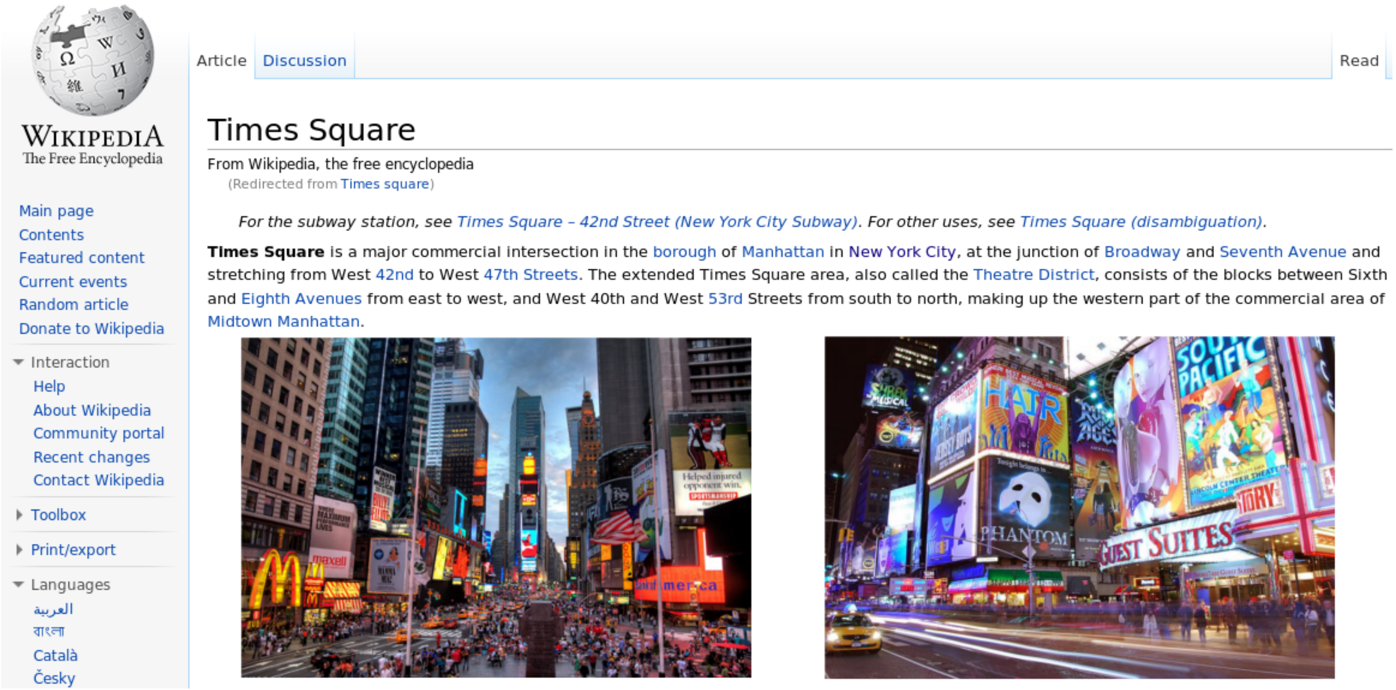
. . . introduces a semantic-based frequency normalization.

Titles of articles are highly expressive—and can be found on the Web.

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

Wikipedia article on “Time Square”:


WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
العربية
বাংলা
Català
Česky

Article | Discussion | Read

Times Square

From Wikipedia, the free encyclopedia

(Redirected from Times square)

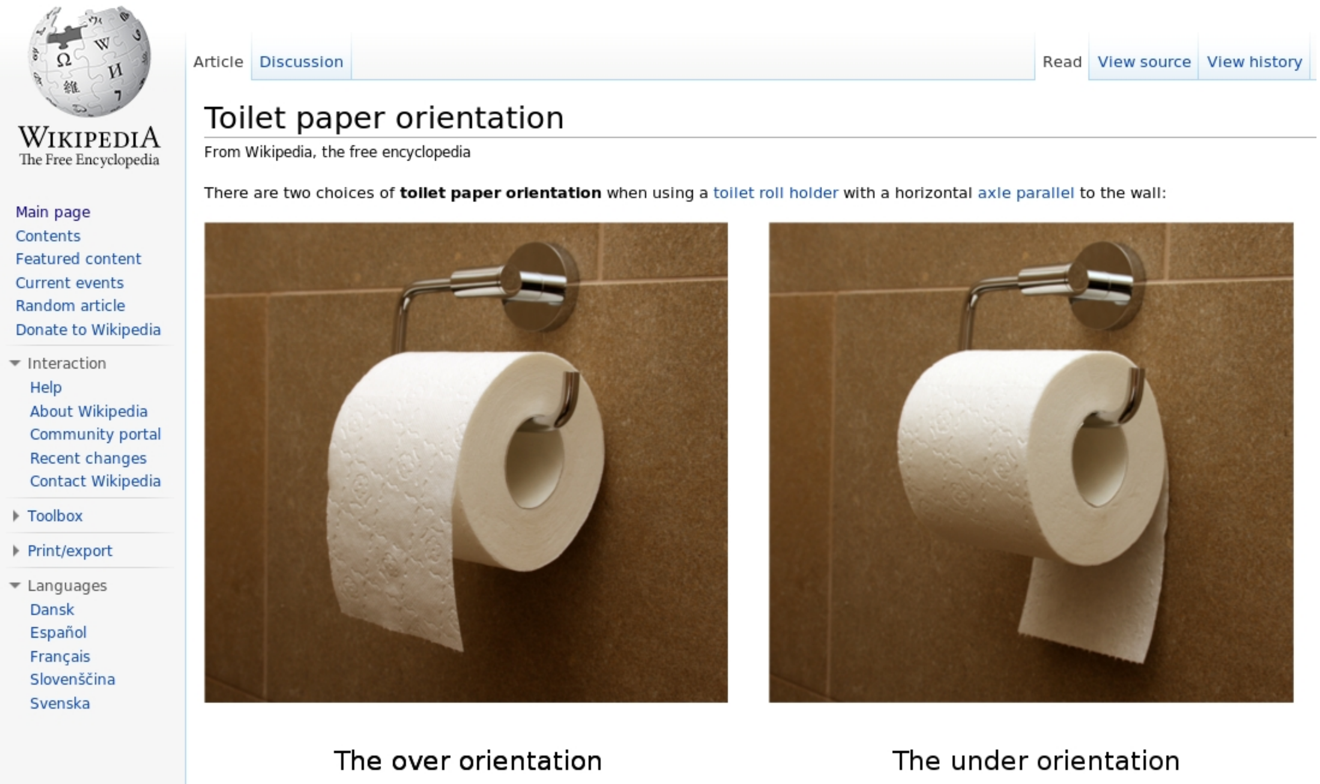
For the subway station, see Times Square – 42nd Street (New York City Subway). For other uses, see Times Square (disambiguation).

Times Square is a major commercial intersection in the borough of Manhattan in New York City, at the junction of Broadway and Seventh Avenue and stretching from West 42nd to West 47th Streets. The extended Times Square area, also called the Theatre District, consists of the blocks between Sixth and Eighth Avenues from east to west, and West 40th and West 53rd Streets from south to north, making up the western part of the commercial area of Midtown Manhattan.

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

Wikipedia article on “Toilet paper orientation”:



Pure regression-based approaches will fail in cases like this.

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

Examples:

segment s	*freq*(s)
`new york`	165.4 million
`new york times`	17.5 million
`new york times square`	20 476
`new york times square dance`	0
`york times`	17.6 million
`york times square`	20 561
`york times square dance`	0
`times square`	1.3 million
`times square dance`	104
`square dance`	210 440

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

Examples:

segment s	*freq*(s)	*wikiTitle*(s)
`new york`	165.4 million	✓
`new york times`	17.5 million	✓
`new york times square`	20 476	-
`new york times square dance`	0	-
`york times`	17.6 million	-
`york times square`	20 561	-
`york times square dance`	0	-
`times square`	1.3 million	✓
`times square dance`	104	-
`square dance`	210 440	✓

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

Examples:

segment s	*freq*(s)	*wikiTitle*(s)	*wikiFreq*(s)
`new york`	165.4 million	✓	165.4 million
`new york times`	17.5 million	✓	165.4 million
`new york times square`	20 476	-	20 476
`new york times square dance`	0	-	0
`york times`	17.6 million	-	17.6 million
`york times square`	20 561	-	20 561
`york times square dance`	0	-	0
`times square`	1.3 million	✓	1.3 million
`times square dance`	104	-	104
`square dance`	210 440	✓	210 440

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

Examples:

segment s	*freq*(s)	*wikiTitle*(s)	*wikiFreq*(s)	$\lvert s\rvert \cdot$ *wikiFreq*(s)
`new york`	165.4 million	✓	165.4 million	330.8 million
`new york times`	17.5 million	✓	165.4 million	496.2 million
`new york times square`	20 476	-	20 476	81 904
`new york times square dance`	0	-	0	0
`york times`	17.6 million	-	17.6 million	35.2 million
`york times square`	20 561	-	20 561	61 683
`york times square dance`	0	-	0	0
`times square`	1.3 million	✓	1.3 million	2.6 million
`times square dance`	104	-	104	312
`square dance`	210 440	✓	210 440	420 880

$$\textit{wikiFreq}(s) = \textit{freq}(s') \quad \leftrightarrow \quad s' \sqsubset s \;\wedge\; \textit{wikiTitle}(s)$$

with $\sqsubset$ as subsequence operator

Query Segmentation

Exploiting Web Phrase Frequencies + Wikipedia Titles

The normalized segment frequencies sum up to the *score* of a query S:

$$\textit{score}(S) = \begin{cases} \displaystyle\sum_{s \in S,\ |s| \geq 2} |s| \cdot \textit{wikiFreq}(s) & \\ -1 & \text{if } |s| \geq 2 \wedge \textit{freq}(s) = 0 \text{ for some } s \in S \end{cases}$$

Ranking:

rank	trend	segmentation S	$\textit{score}(S)$
1	$\uparrow\uparrow$	`"new york times" "square dance"`	496.6 million
2	$\uparrow\uparrow$	`"new york times" square dance`	496.2 million
3	$\downarrow$	`"new york" "times square" dance`	333.4 million
⋮	⋮	⋮	⋮
13	-	`new york "times square dance"`	312
14	-	`new york times square dance`	0
15	-	`"new york times square dance"`	-1
16	-	`new "york times square dance"`	-1

Query Segmentation

 $\vee$ $\vee$ $\ldots$

How do our approaches perform?

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effectiveness is measured:

Reference:	`"new york" "times square" dance`	(three segments)
Computed:	`"new york" times square dance`	(four segments)

→ Query: 0 (computed $\neq$ reference)

→ Precision: $\frac{1}{2}$ (2 out of 4 computed segments correct)

→ Recall: $\frac{2}{3}$ (2 out of 3 reference segments found)

→ Break: $\frac{3}{4}$ (3 out of 4 between-words decisions correct)

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effectiveness is measured:

Reference:	`"new york" "times square" dance`	(three segments)
Computed:	`"new york" times square dance`	(four segments)

→ Query: 0 (computed $\neq$ reference)

→ Precision: $\frac{1}{2}$ (2 out of 4 computed segments correct)

→ Recall: $\frac{2}{3}$ (2 out of 3 reference segments found)

→ Break: $\frac{3}{4}$ (3 out of 4 between-words decisions correct)

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effectiveness is measured:

Reference:	`"new york" "times square" dance`	(three segments)
Computed:	`"new york" times square dance`	(four segments)

→ Query: 0 (computed $\neq$ reference)

→ Precision: $\frac{1}{2}$ (2 out of 4 computed segments correct)

→ Recall: $\frac{2}{3}$ (2 out of 3 reference segments found)

→ Break: $\frac{3}{4}$ (3 out of 4 between-words decisions correct)

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effectiveness is measured:

Reference:	`"new york" "times square" dance`	(three segments)
Computed:	`"new york" times square dance`	(four segments)

→ Query: 0 (computed $\neq$ reference)

→ Precision: $\frac{1}{2}$ (2 out of 4 computed segments correct)

→ Recall: $\frac{2}{3}$ (2 out of 3 reference segments found)

→ Break: $\frac{3}{4}$ (3 out of 4 between-words decisions correct)

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effectiveness is measured:

Reference:	`"new york" "times square" dance`	(three segments)
Computed:	`"new|york"| times square |dance`	(four segments)

→ Query: 0 (computed $\neq$ reference)

→ Precision: $\frac{1}{2}$ (2 out of 4 computed segments correct)

→ Recall: $\frac{2}{3}$ (2 out of 3 reference segments found)

→ Break: $\frac{3}{4}$ (3 out of 4 between-words decisions correct)

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effective we are:

	Mutual information	Bergsma/Wang	$\lvert s\rvert^{\lvert s\rvert} \cdot \mathit{freq}(s)$	$\lvert s\rvert \cdot \mathit{wikiFreq}(s)$
Query	0.583	0.702	0.700	**0.726**
Precision	0.693	0.812	0.800	**0.820**
Recall	0.697	**0.831**	0.796	0.807
Break	0.849	0.899	0.889	**0.900**

Query Segmentation

About Effectiveness

The standard corpus: [Bergsma and Wang, EMNLP-CoNLL 2007]

- 500 queries from the AOL log
- each segmented by three human annotators
- often used for evaluation

How effective we are:

	Mutual information	Bergsma/Wang	$\lvert s\rvert^{\lvert s\rvert} \cdot$ *freq*(s)	$\lvert s\rvert \cdot$ *wikiFreq*(s)
Query	0.583	0.702	0.700	**0.726**
Precision	0.693	0.812	0.800	**0.820**
Recall	0.697	**0.831**	0.796	0.807
Break	0.849	0.899	0.889	**0.900**

Shortcomings of the Bergsma/Wang-corpus:

- not representative (small, just noun-phrases)
- only three annotators (40% without majority)
- duplicates, typos, encoding errors

Query Segmentation

About Effectiveness

A new evaluation corpus: [Hagen et al., WWW 2011]

- 50 000 queries (3-10 keywords) from "filtered" AOL log
- sampling considers frequency and length distribution
- semi-automatic spell checking (14% corrected)
- 10 annotators per query via Amazon Mechanical Turk

How effective we are:

	Mutual information	$\lvert s\rvert^{\lvert s\rvert} \cdot \mathit{freq}(s)$	$\lvert s\rvert \cdot \mathit{wikiFreq}(s)$
Query	0.598	0.599	**0.616**
Precision	0.727	0.736	**0.744**
Recall	0.738	0.733	**0.739**
Break	0.844	0.842	**0.850**

Query Segmentation

About Efficiency

System and implementation details:

- standard quad-core PC running Ubuntu 10.04
- hash table (MPHF) for about 2 billion normalized frequencies
- 12 GB memory footprint

Throughput:

- > 3 000 queries per second
- Remark: 1 billion queries per day means 12 000 queries per second

Query Segmentation

Query Segmentation Summary

What we have done:

- exploitation of the Google n-gram corpus
- regression-based $|s|^{|s|}$-normalization strategy
- Wikipedia-based normalization strategy
- as effective as state of the art, but more robust and faster
- new evaluation corpus (about two orders of magnitude larger than previous STA)

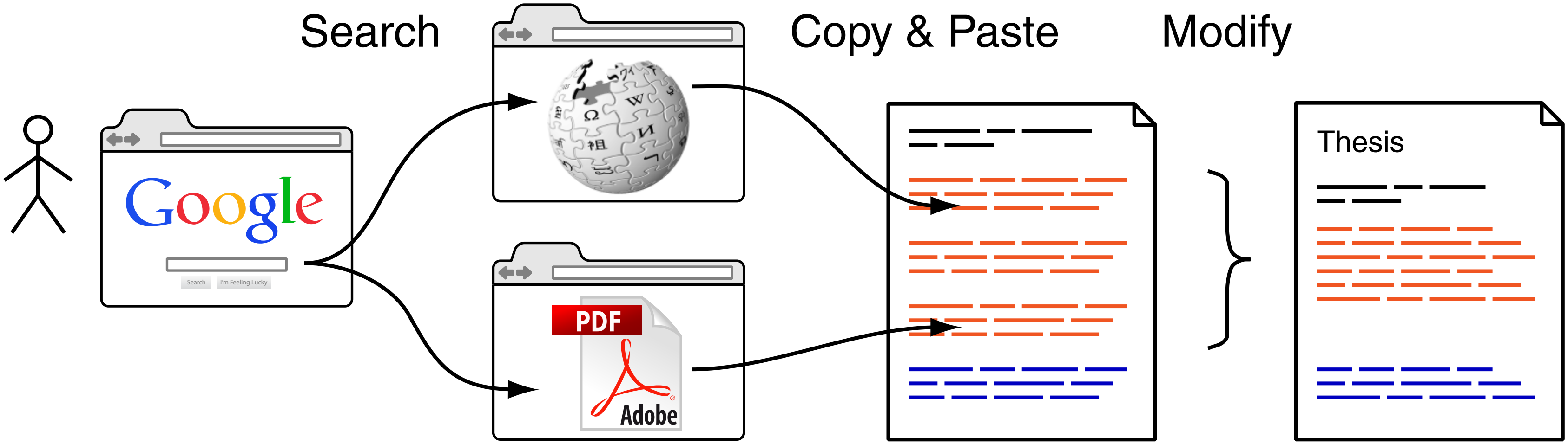
What we plan to do:

- *ranking-aware* effectiveness
- *retrieval-aware* effectiveness

Candidate Retrieval at PAN'12

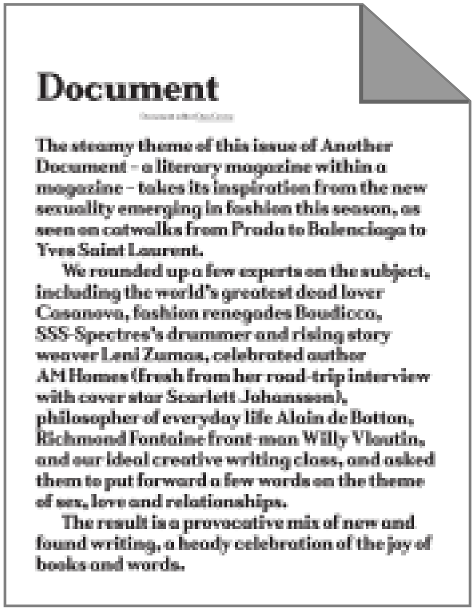
Candidate Retrieval at PAN'12

How Humans Plagiarize



Candidate Retrieval at PAN'12

How Humans Spot Plagiarism

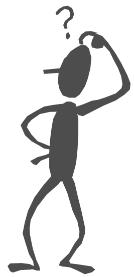


Document

The steamy theme of this issue of Another Document – a literary magazine within a magazine – takes its inspiration from the new sexuality emerging in fashion this season, as seen on catwalks from Prada to Balenciaga to Yves Saint Laurent.

We rounded up a few experts on the subject, including the world's greatest dead lover Casanova, fashion renegades Boudicca, SSS-Spectres's drummer and rising story weaver Leni Zumas, celebrated author AM Homes (fresh from her road-trip interview with cover star Scarlett Johansson), philosopher of everyday life Alain de Botton, Richmond Fontaine front-man Willy Vlautin, and our ideal creative writing class, and asked them to put forward a few words on the theme of sex, love and relationships.

The result is a provocative mix of new and found writing, a heady celebration of the joy of books and words.



Candidate Retrieval at PAN'12

How Humans Spot Plagiarism

Candidate Retrieval at PAN'12

Algorithmic Plagiarism Detection

Keyword extraction from the document	Candidate retrieval in the WWW	Detailed comparison	Knowledge-based post-processing
Step 1	Step 2	Step 3	Step 4

Candidate Retrieval at PAN'12

Algorithmic Plagiarism Detection

Step 1	Step 2	Step 3	Step 4
Keyword extraction from the document	Candidate retrieval in the WWW	Detailed comparison	Knowledge-based post-processing

Where are the crucial keywords?

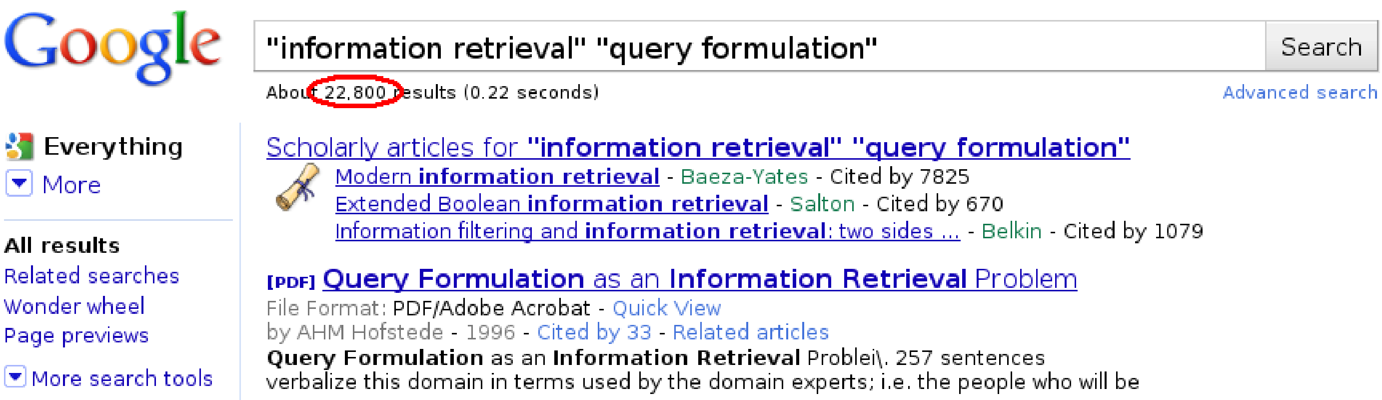
- check for noun phrases
- find orthographic mistakes
- consider word frequency classes
- but—don't look in titles, captions, or headings

Candidate Retrieval at PAN'12

Algorithmic Plagiarism Detection

Step 1	Step 2	Step 3	Step 4
Keyword extraction from the document	Candidate retrieval in the WWW	Detailed comparison	Knowledge-based post-processing

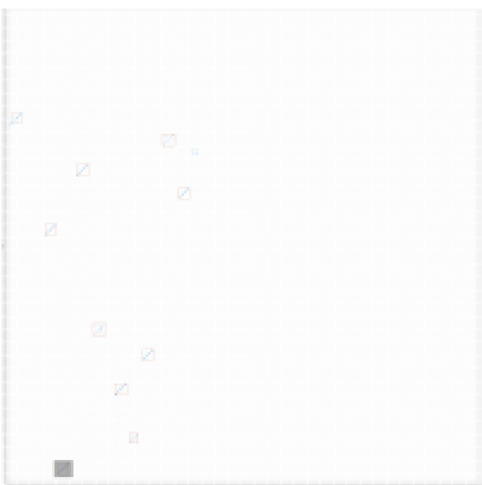
Keywords: "information retrieval", "query formulation", "search session", "user support"



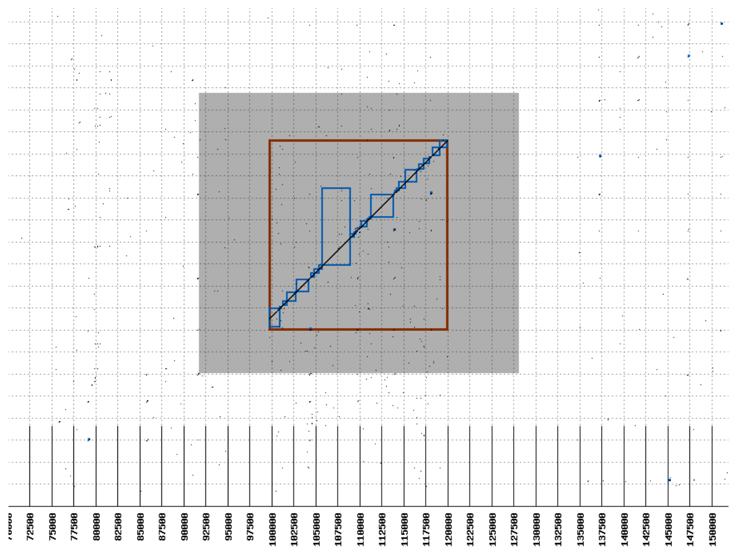
Candidate Retrieval at PAN'12

Algorithmic Plagiarism Detection

Keyword extraction from the document	Candidate retrieval in the WWW	Detailed comparison	Knowledge-based post-processing
Step 1	Step 2	Step 3	Step 4



$\rightarrow \ \ldots \ \rightarrow$



Candidate Retrieval at PAN'12

Algorithmic Plagiarism Detection

Step 1	Step 2	Step 3	Step 4
Keyword extraction from the document	Candidate retrieval in the WWW	Detailed comparison	Knowledge-based post-processing

Check for problematic decisions:

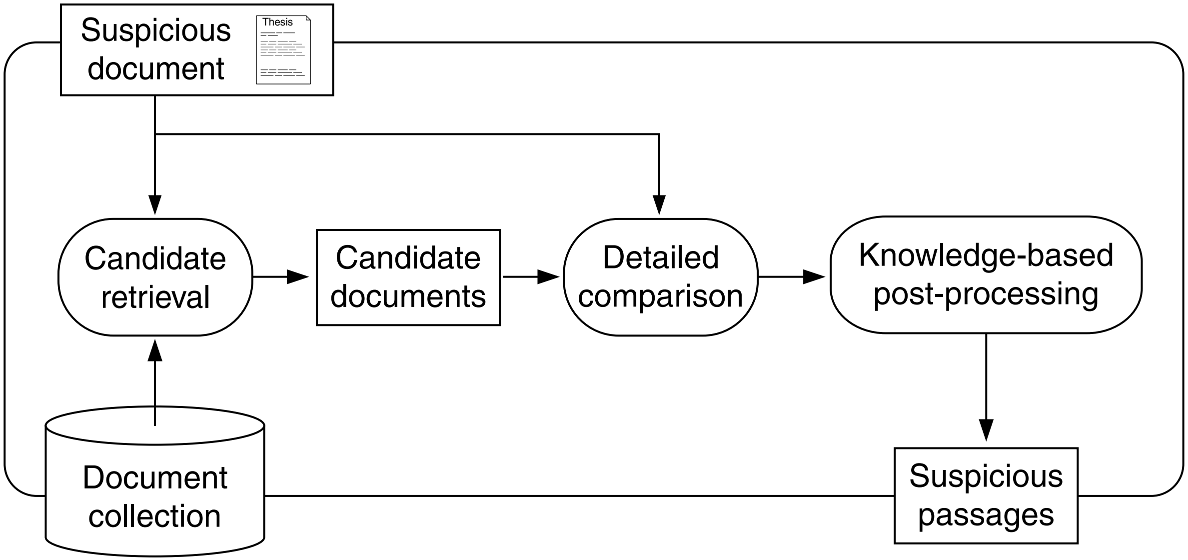
- citation analysis

  (difficult: consider "excuse citations" in footnotes along with a completely reused text)

- comparison of authors and co-authors
- visualization

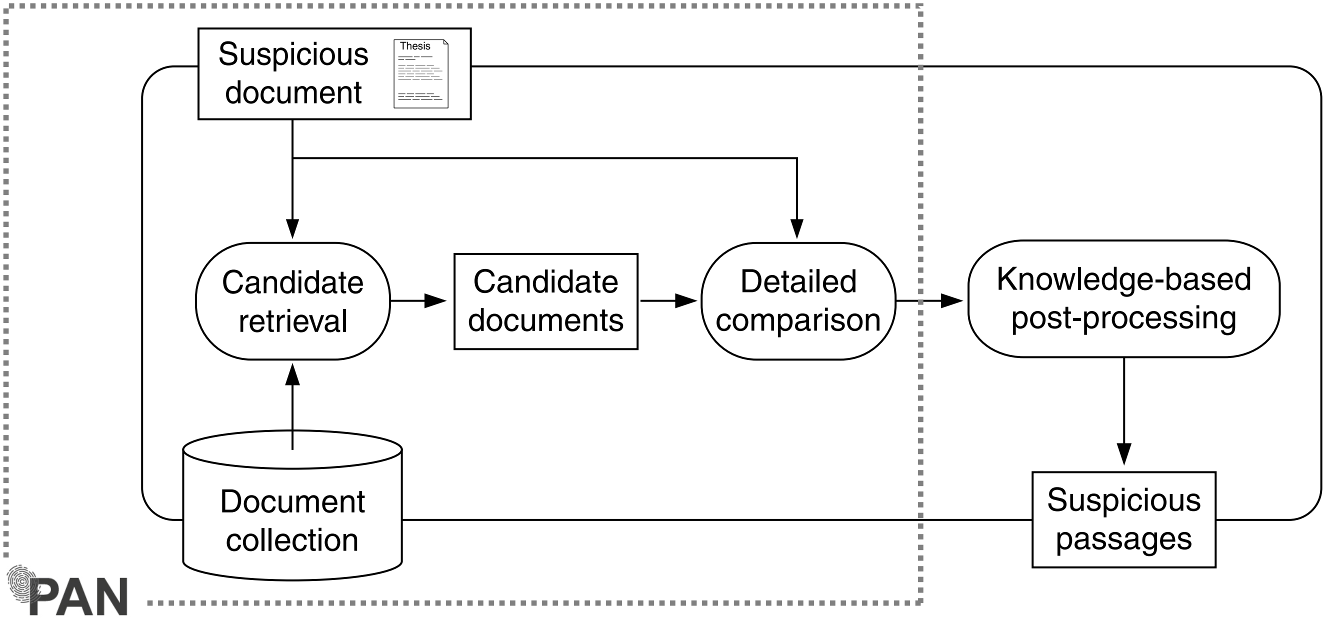
Candidate Retrieval at PAN'12

Algorithmic Plagiarism Detection



Candidate Retrieval at PAN'12

PAN Campaign [pan.webis.de]

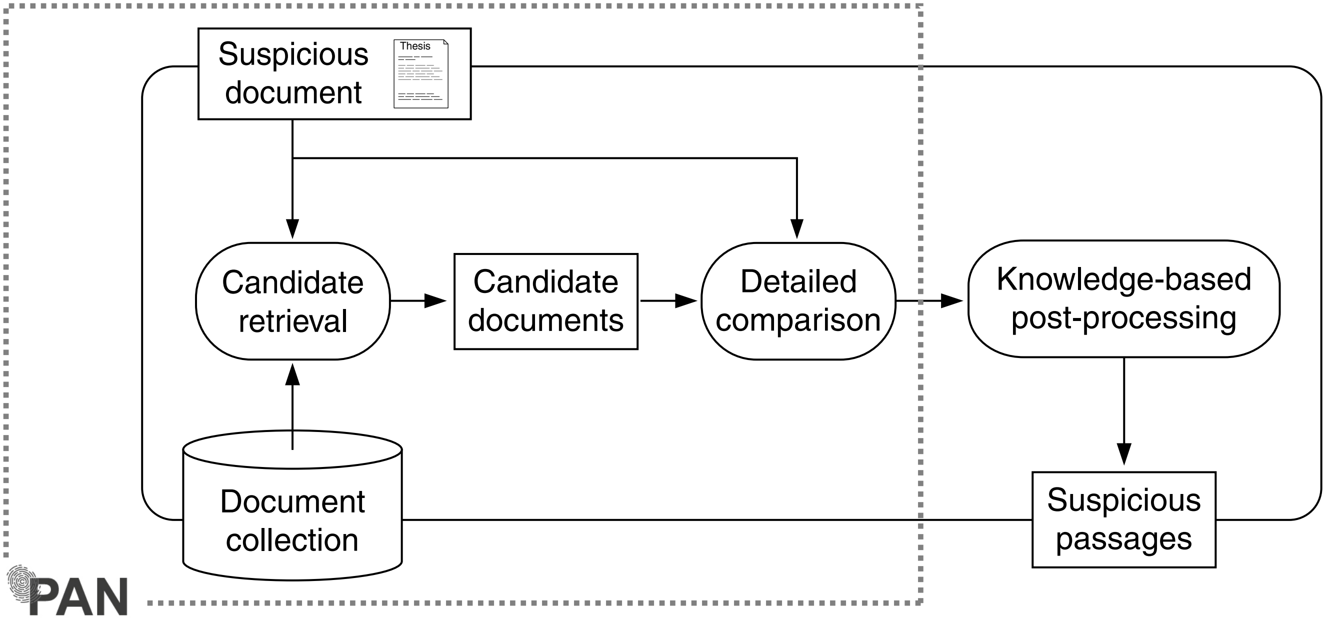


Uncovering plagiarism, authorship, and social software misuse:

- initiated in 2007
- competitions since 2009 (detection of authorship, vandalism, plagiarism, etc.)
- hosted at SIGIR, ECAI, SEPLN, and CLEF (since 2010)
- regularly >10 groups who participate in the plagiarism detection task

Candidate Retrieval at PAN'12

PAN Campaign

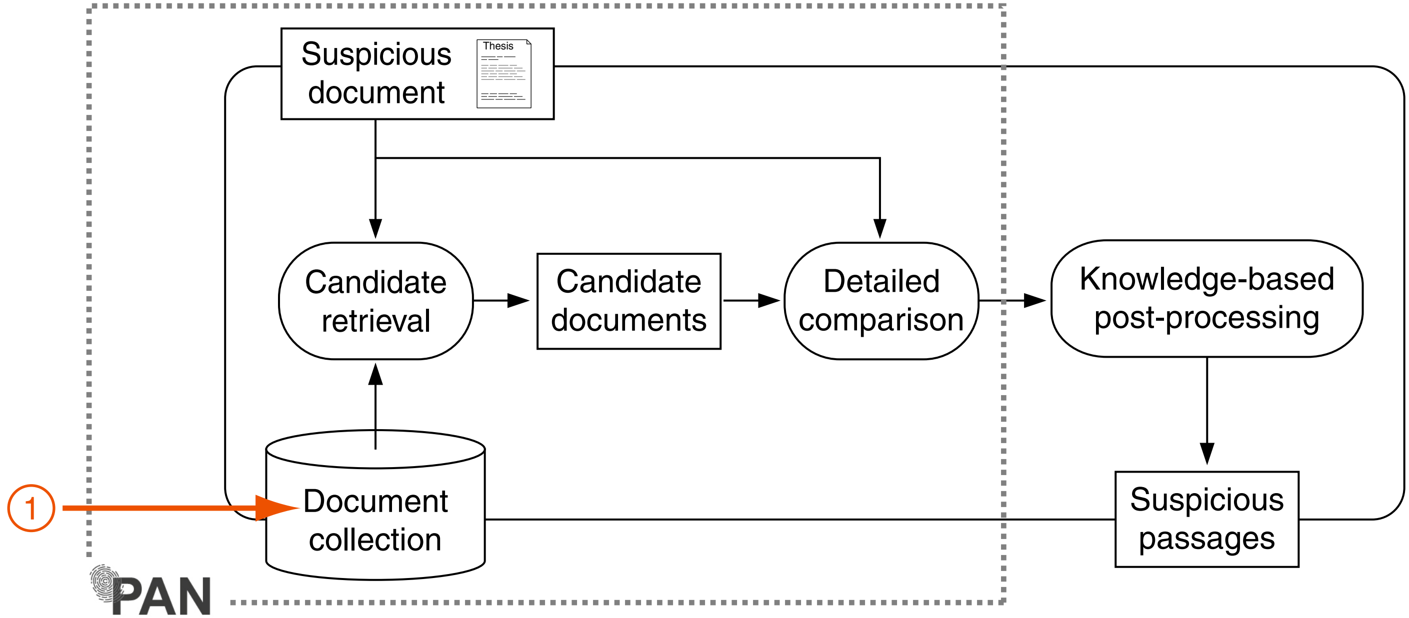


Observations, problems:

1. PAN-PC-10 corpus based on 27 073 documents, contains 68 558 plagiarism cases, addresses a broad range (by varying length, paraphrasing, topic alignment, etc.).
2. But—the corpus is too small to enforce a true candidate retrieval situation: most participants did a complete detailed comparison on all $O(n^2)$ document pairs.
3. Corpora quality issues: plagiarized passages consider not the surrounding document, paraphrasing mostly done by machines, the Web is not used as source.

Candidate Retrieval at PAN'12

PAN Campaign



Observations, problems:

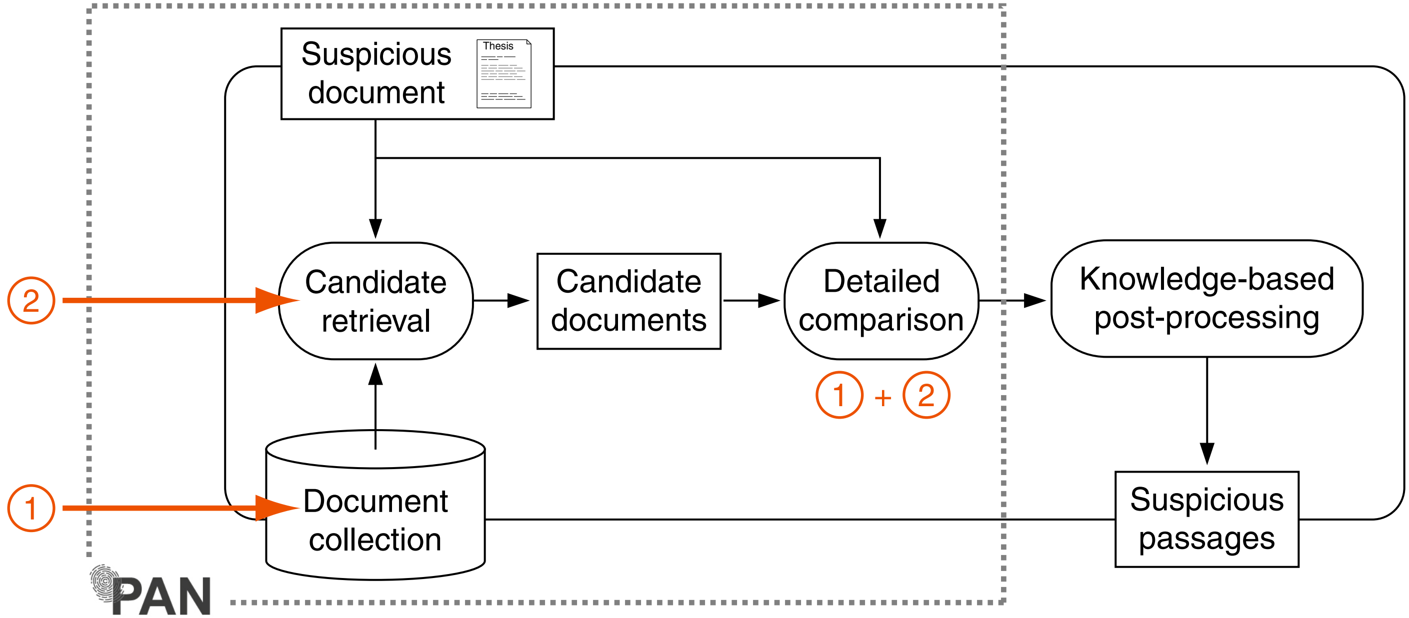
1. PAN-PC-10 corpus based on 27 073 documents, contains 68 558 plagiarism cases, addresses a broad range (by varying length, paraphrasing, topic alignment, etc.).

2. But—the corpus is too small to enforce a true candidate retrieval situation: most participants did a complete detailed comparison on all $O(n^2)$ document pairs.

3. Corpora quality issues: plagiarized passages consider not the surrounding document, paraphrasing mostly done by machines, the Web is not used as source.

Candidate Retrieval at PAN'12

PAN Campaign

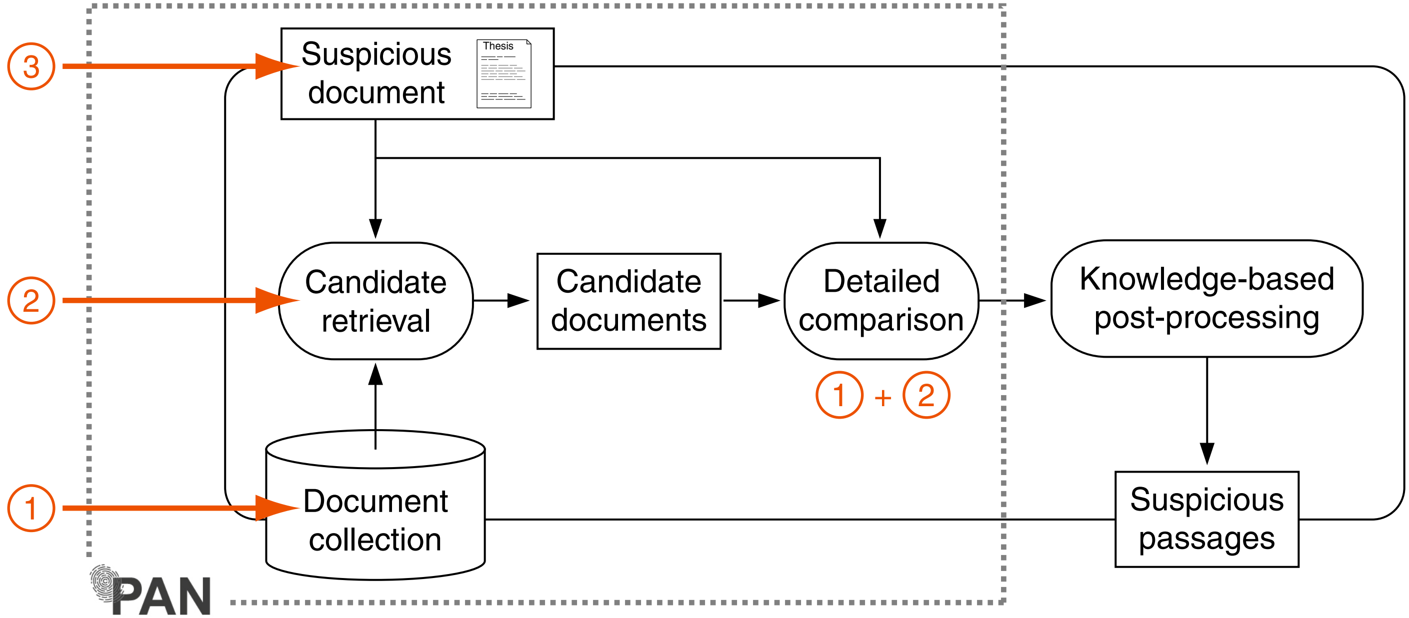


Observations, problems:

1. PAN-PC-10 corpus based on 27 073 documents, contains 68 558 plagiarism cases, addresses a broad range (by varying length, paraphrasing, topic alignment, etc.).
2. But—the corpus is too small to enforce a true candidate retrieval situation: most participants did a complete detailed comparison on all $O(n^2)$ document pairs.
3. Corpora quality issues: plagiarized passages consider not the surrounding document, paraphrasing mostly done by machines, the Web is not used as source.

Candidate Retrieval at PAN'12

PAN Campaign

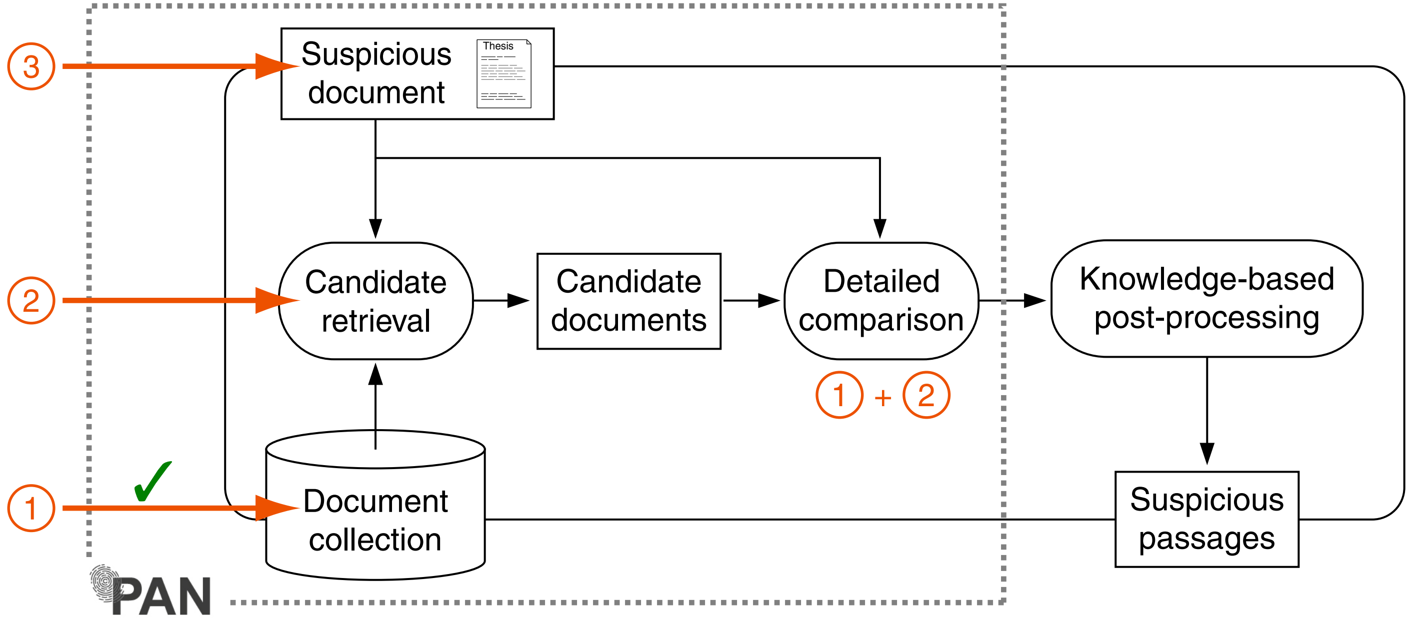


Observations, problems:

1. PAN-PC-10 corpus based on 27 073 documents, contains 68 558 plagiarism cases, addresses a broad range (by varying length, paraphrasing, topic alignment, etc.).
2. But—the corpus is too small to enforce a true candidate retrieval situation: most participants did a complete detailed comparison on all $O(n^2)$ document pairs.
3. Corpora quality issues: plagiarized passages consider not the surrounding document, paraphrasing mostly done by machines, the Web is not used as source.

Candidate Retrieval at PAN'12

PAN Campaign

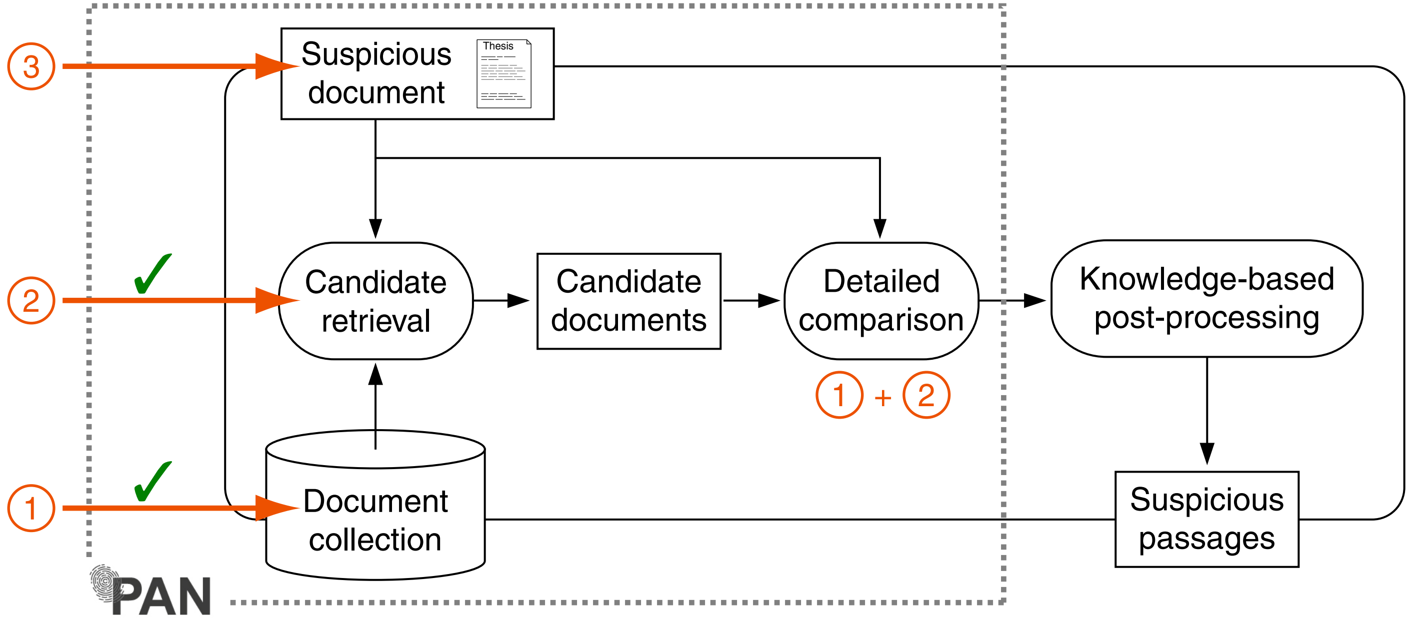


Considerations:

1. PAN'12 will use the English part of the ClueWeb09 corpus (used in TREC 2009-2011 for several tracks) as a static Web snapshot. Size: 500 million web pages, 12.5TB
2. Participants get efficient corpus access via the API of the ChatNoir search engine. ClueWeb and ChatNoir will ensure experiment reproducibility and controllability.
3. The new corpus: manually written digestible texts, topically matching plagiarism cases, Web as source (for document synthesis and plagiarism detection).

Candidate Retrieval at PAN'12

PAN Campaign



Considerations:

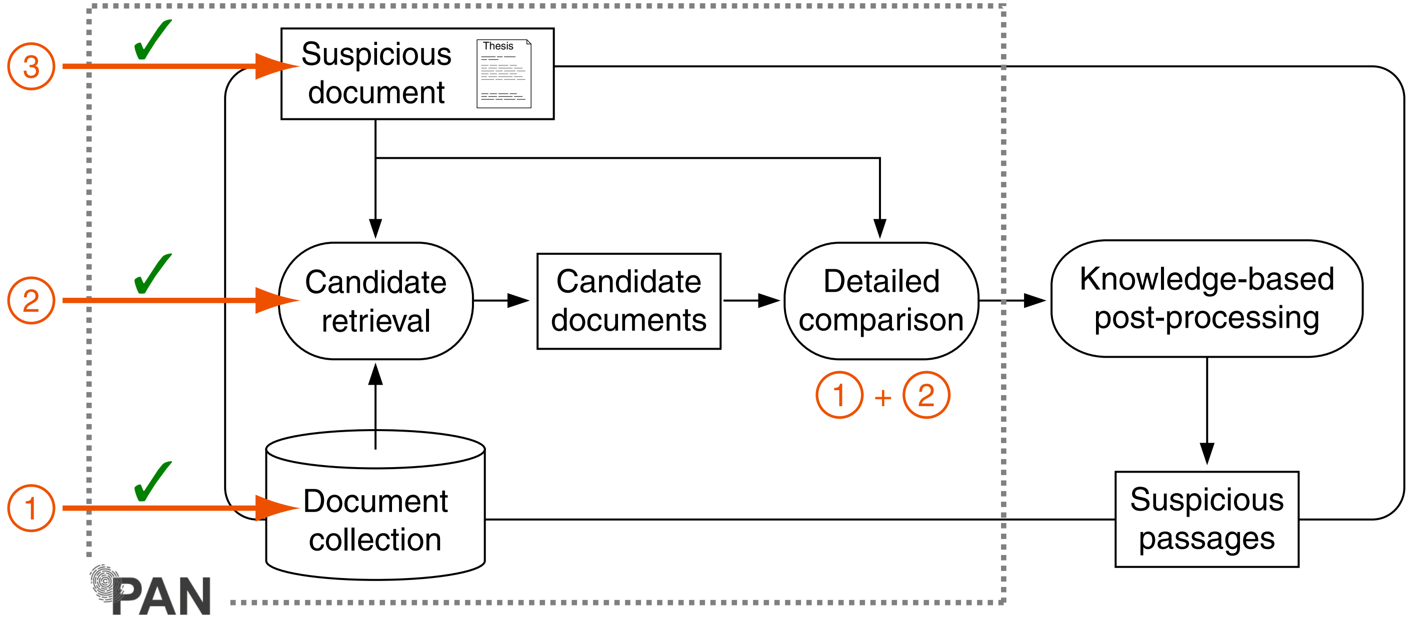
1. PAN'12 will use the English part of the ClueWeb09 corpus (used in TREC 2009-2011 for several tracks) as a static Web snapshot. Size: 500 million web pages, 12.5TB

2. Participants get efficient corpus access via the API of the ChatNoir search engine. ClueWeb and ChatNoir will ensure experiment reproducibility and controllability.

3. The new corpus: manually written digestible texts, topically matching plagiarism cases, Web as source (for document synthesis and plagiarism detection).

Candidate Retrieval at PAN'12

PAN Campaign



Considerations:

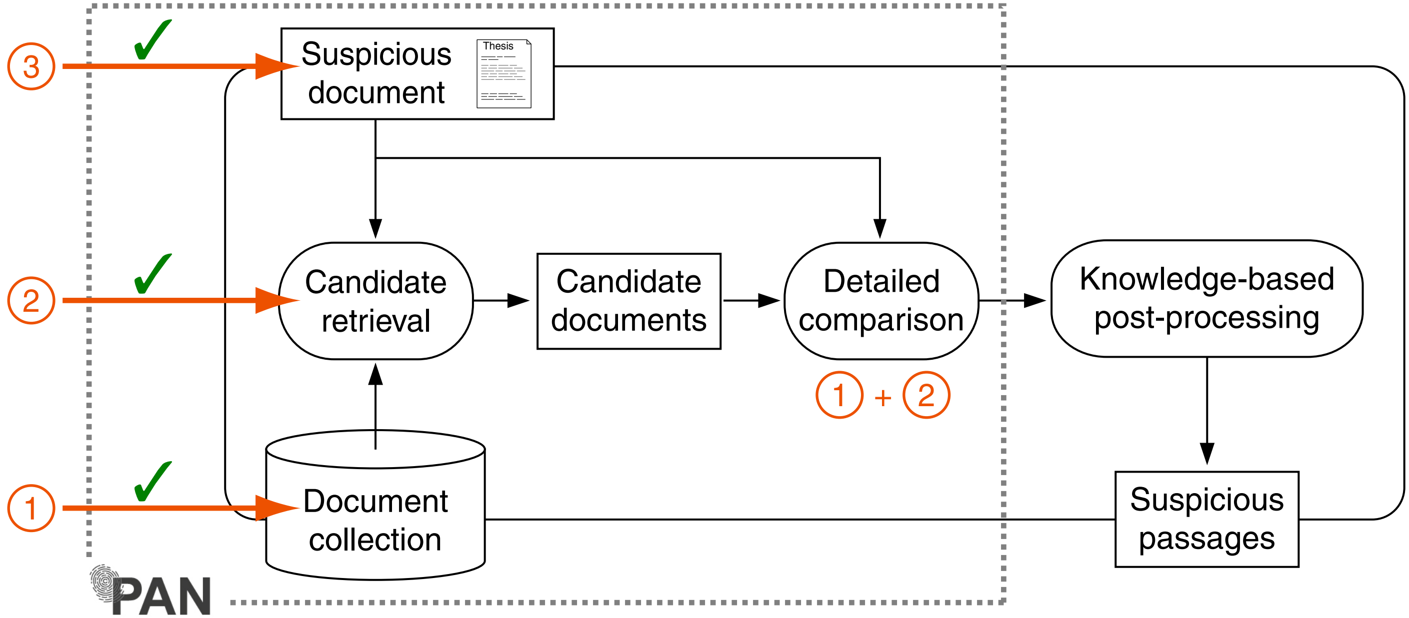
1. PAN'12 will use the English part of the ClueWeb09 corpus (used in TREC 2009-2011 for several tracks) as a static Web snapshot. Size: 500 million web pages, 12.5TB

2. Participants get efficient corpus access via the API of the ChatNoir search engine. ClueWeb and ChatNoir will ensure experiment reproducibility and controllability.

3. The new corpus: manually written digestible texts, topically matching plagiarism cases, Web as source (for document synthesis and plagiarism detection).

Candidate Retrieval at PAN'12

PAN Campaign

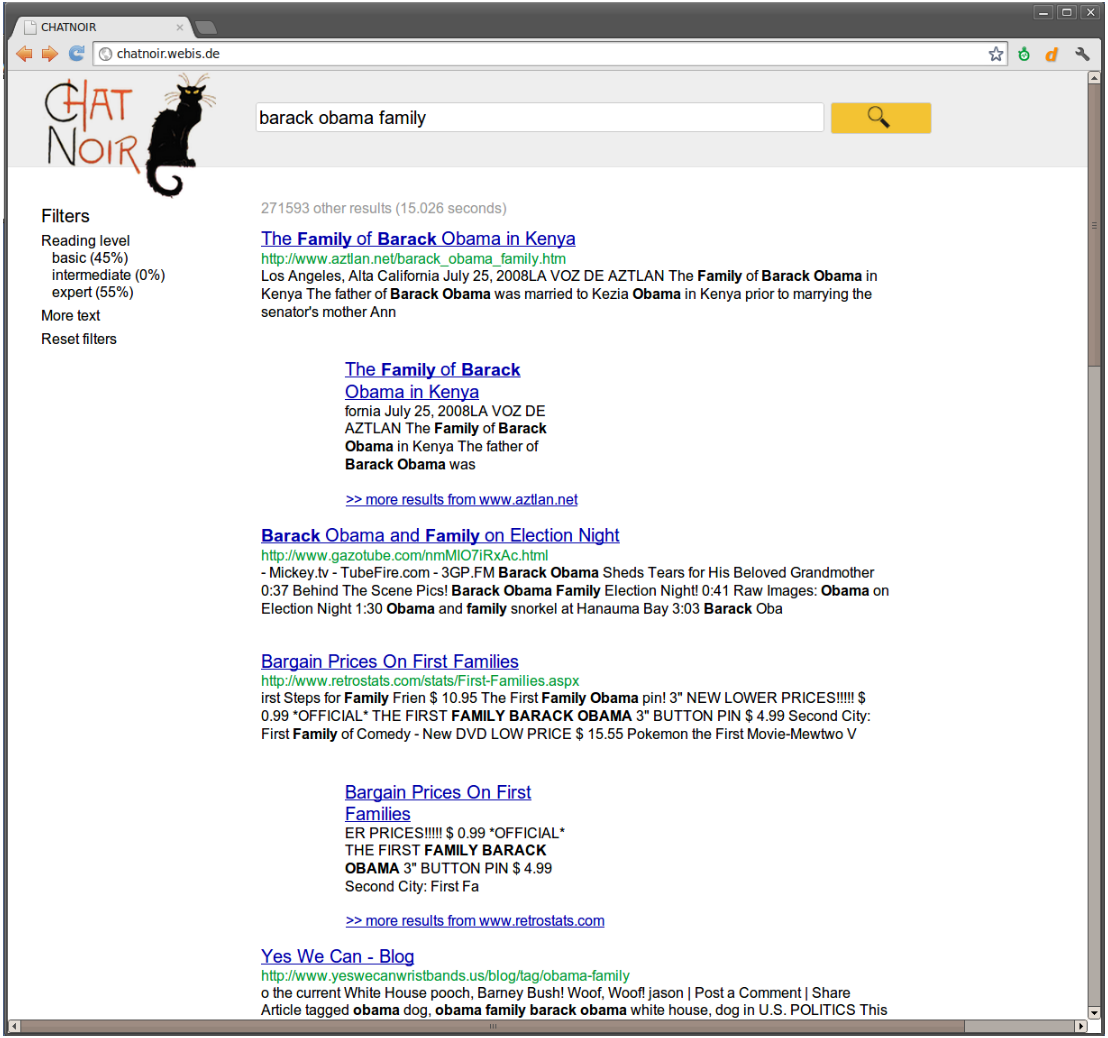


PAN competition 2012: [pan.webis.de]

- Key intention corpus: humans write essays on given topics, plagiarizing from the ClueWeb, using the ChatNoir search engine.
- Key intention competition: detectors use ChatNoir to retrieve candidate documents from the ClueWeb, using the ChatNoir search engine.
- Constraint: detectors get a budget of queries to model the cost scheme of commercial search engine APIs wrt. posed queries and downloaded documents.

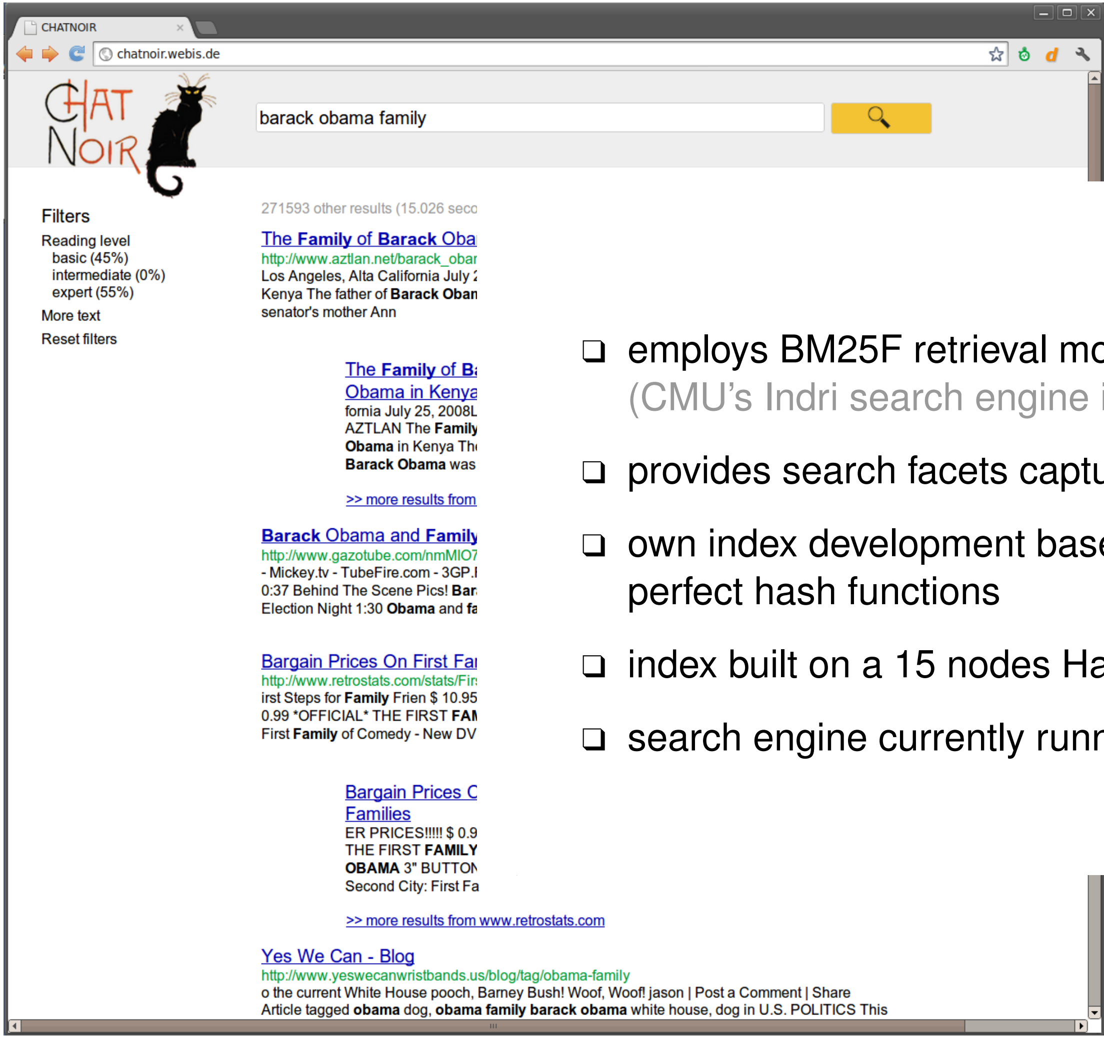
Candidate Retrieval at PAN'12

About ChatNoir [chatnoir.webis.de]



Candidate Retrieval at PAN'12

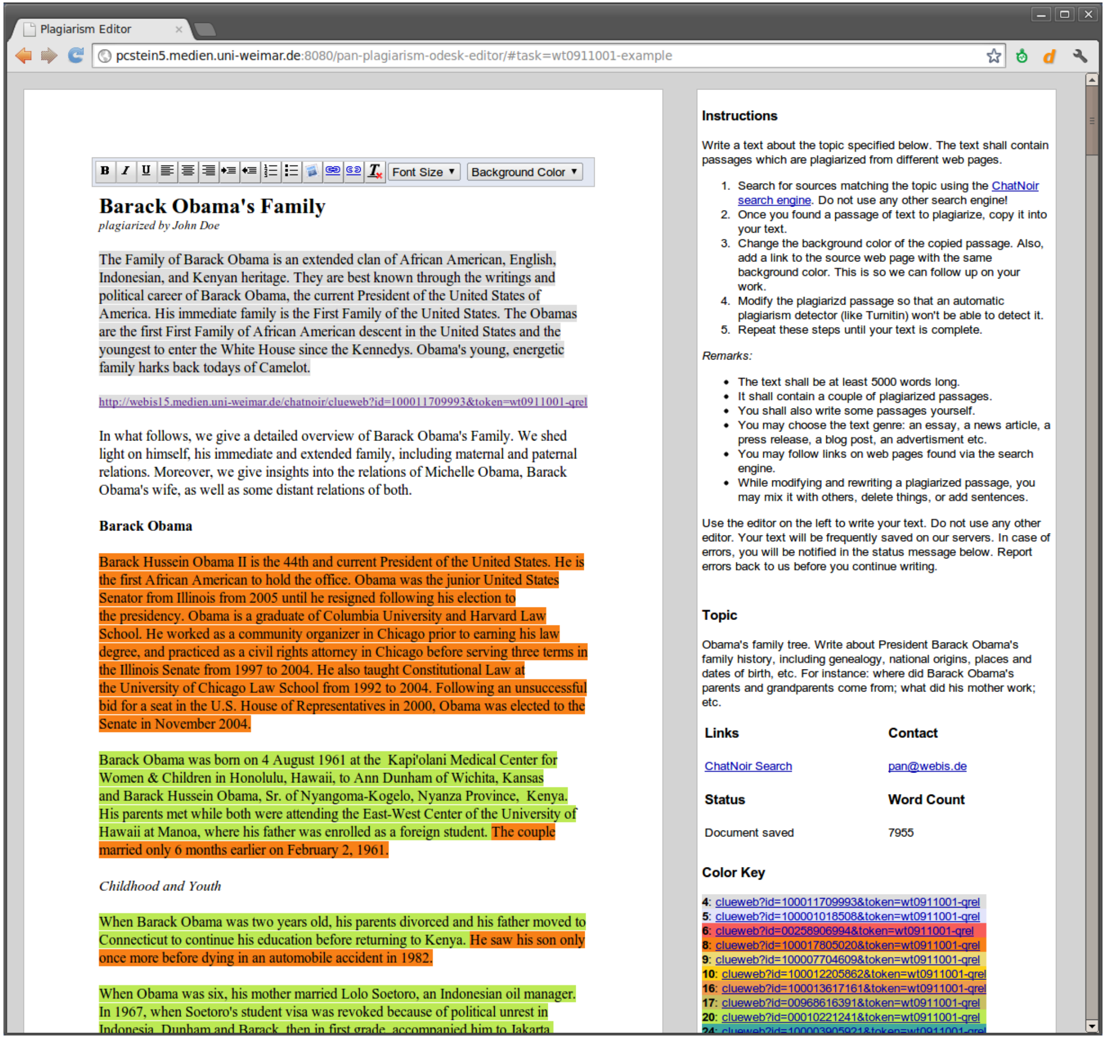
About ChatNoir [chatnoir.webis.de]



- employs BM25F retrieval model
  (CMU's Indri search engine is language-model-based)
- provides search facets capturing readability issues
- own index development based on externalized minimum perfect hash functions
- index built on a 15 nodes Hadoop cluster
- search engine currently running on 11 machines

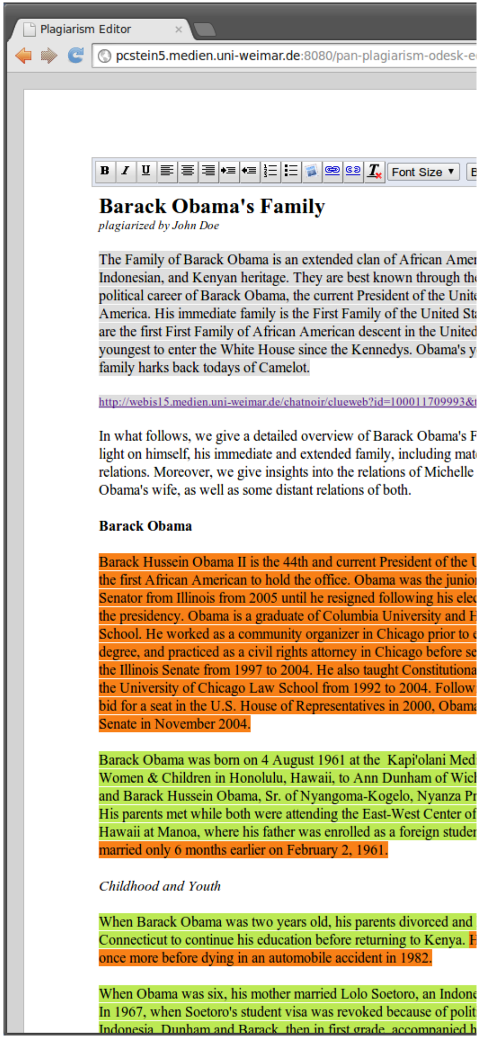
Candidate Retrieval at PAN'12

About Corpus Construction



Candidate Retrieval at PAN'12

About Corpus Construction



- ❑ an essays has approx. 5000 words which means 8-10 pages
- ❑ corpus will be freely available after the competition
- ❑ own web editor was developed for essay writing
- ❑ the writing is crowdsourced via oDesk

➜ full control over:
  - plagiarized document
  - anonymized author identifier
  - set of used source documents
  - annotations of paraphrased passages
  - query log of the writer while writing the text
  - search results for each query
  - click-through data for each query
  - browsing data of links clicked within ClueWeb
  - key log of the document covering all keystrokes
  - work diary and screenshots as recorded by oDesk

➜ insights on how humans work when reusing text

TIRA: Experiments as a Service [tira.webis.de]

Retrieve research results along with their experiments.

Candidate Retrieval at PAN'12

TIRA: Experiments as a Service

TIRA takes a locally executable program and turns it into a web service. The TIRA approach features:

- Local execution
- Platform independence
- Result / experiment retrieval
- Web dissemination
- Peer-to-peer collaboration

Candidate Retrieval at PAN'12

TIRA: Experiments as a Service

TIRA takes a locally executable program and turns it into a web service. The TIRA approach features:

- Local execution
  Data is kept confidential; the framework can reside with the data.
- Platform independence
  A programs can be deployed as a web service without modifications.
- Result / experiment retrieval
  Result management and online retrieval of matching experiments.
- Web dissemination
  Experiments can be cited through their unique URL in publications.
- Peer-to-peer collaboration
  TIRA instances can be connected to a network of experimentation nodes.

Candidate Retrieval at PAN'12

TIRA: Experiments as a Service

PAN'12 comes with a TIRA experimentation service for evaluating and communicating the participants' training results.

tιpα PAN12 -- Evaluation Service: Training Phase

Welcome to the PAN12 evaluation service.
In the form below, you can upload your detection results for the six datasets in the training corpus. The service expects the detection XML files you created for each pair of suspicious and source document as a zip file. Use a constant team name for your submissions to make your results easily retrievable with the Search button. When every field is filled in, press Execute and find your performance results in the result table below.

- Team Name: baseline
- Training Dataset: -
- Detection Zip: [Upload File]

[Search] [Execute]

Team Name	Training Dataset	Detection Zip	PlagDet	Precision	Recall	Granul.	Result	Status
baseline	All datasets together	$UPLOAD/9	0.125	0.978	0.125	2.446	scores	DONE
baseline	01_no_plagiarism	$UPLOAD/10	1.0	1.0	1.0	1.0	scores	DONE
baseline	02_no_obfuscation	$UPLOAD/11	0.860	0.861	0.860	1.001	scores	DONE
baseline	03_artificial_low	$UPLOAD/12	0.100	0.997	0.111	2.997	scores	DONE
baseline	04_artificial_high	$UPLOAD/13	0.005	0.999	0.002	1.077	scores	DONE
baseline	05_translation	$UPLOAD/14	0.000	1.0	0.000	1.214	scores	DONE
baseline	06_simulated_paraphrase	$UPLOAD/15	0.056	0.998	0.072	4.307	scores	DONE

Candidate Retrieval at PAN'12

TIRA: Experiments as a Service

All you need to create the shown web page:

1. The generic command to run the PAN evaluation measure:

   ```
   python perfmeasure.py -t $team -p $truth -d $det > scores.txt
   ```

2. The parameter definitions:

   ```
   $team  → [a-zA-Z0-9]+
   $det   → [a-zA-Z0-9]+\.zip
   $truth → 01_no_plagiarism | 02_no_obfuscation |
            03_artificial_low | 04_artificial_high |
            05_translation | 06_simulated_paraphrase
   ```

3. A program description and descriptive labels for the parameters (optional).

Candidate Retrieval at PAN'12

Candidate Retrieval Summary

Key elements of the PAN'12 plagiarism competition:

- high-quality corpus in the form of short essays created by humans
- ClueWeb corpus to create as well as to hide plagiarized texts
- ChatNoir search engine as efficient and open API for retrieval
- TIRA platform for technology comparison and result dissemination

Candidate Retrieval at PAN'12

Candidate Retrieval Summary

Key elements of the PAN'12 plagiarism competition:

- high-quality corpus in the form of short essays created by humans
- ClueWeb corpus to create as well as to hide plagiarized texts
- ChatNoir search engine as efficient and open API for retrieval
- TIRA platform for technology comparison and result dissemination

We are hoping to benefit in the following ways:

- realistic candidate retrieval situation from a
  - (a) plagiarism creation perspective
  - (b) plagiarism detection perspective
  - (c) computation resource perspective
- new search strategies for known-item-finding in the Web
- open and publicly available benchmarks

Almost the End

Almost the End

What We have Seen

- The Netspeak Word Search Engine
- Query Segmentation using the WAC
- Candidate Retrieval at PAN’12

Thank you for listening!