

Moshe Koppel - Jonathan Schler - Shlomo Argamon

Authorship attribution in the wild

FABIAN MÜLLER

Lang Resources & Evaluation (2011) 45:83-94
DOI 10.1007/s10579-009-9111-2

Authorship attribution in the wild

Moshe Koppel · Jonathan Schler · Shlomo Argamon

Published online: 13 January 2010
© Springer Science+Business Media B.V. 2010

Abstract Most previous work on authorship attribution in which we need to identify candidate...

Approach

Experiment

Implementation

Results

Agenda

- the approach proposed by Koppel et al.
- experiment, main algorithm and used data set
- my re-implementation of the experiment
 - code samples
 - encountered problems while implementing
- Results and conclusion



Approach

Experiment

Implementation

Results

Introduction

	Simple problem	Koppel et al.
Candidate authors	small, closed set	thousands
„Known text“	✓ available	✗ might be very limited
Actual author in candidate set	✓ yes	? might be not

Approach

Experiment

Implementation

Results

Introduction

Goals:

- high attribution precision
- acceptable amount of recall
- measure effect of key factors:
 1. number of candidates
 2. size of known-text
 3. size of anonymous text

Koppel et al.

Authors

thousands

Known-text

✗ might be very limited

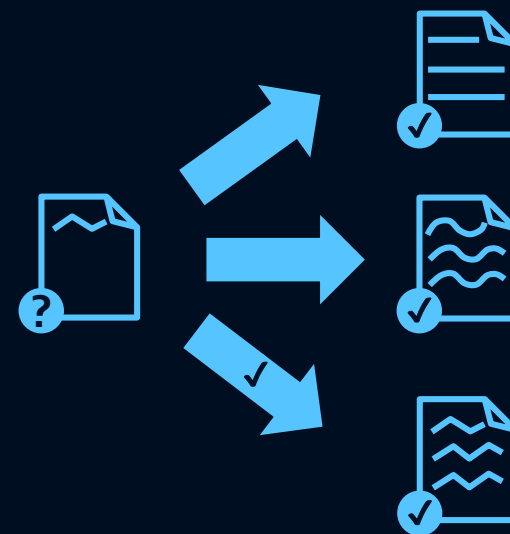
Author in set

? might be not

Methods of authorship attribution

2 main paradigms:

- **Similarity-based** paradigm
- **Machine-learning** paradigm



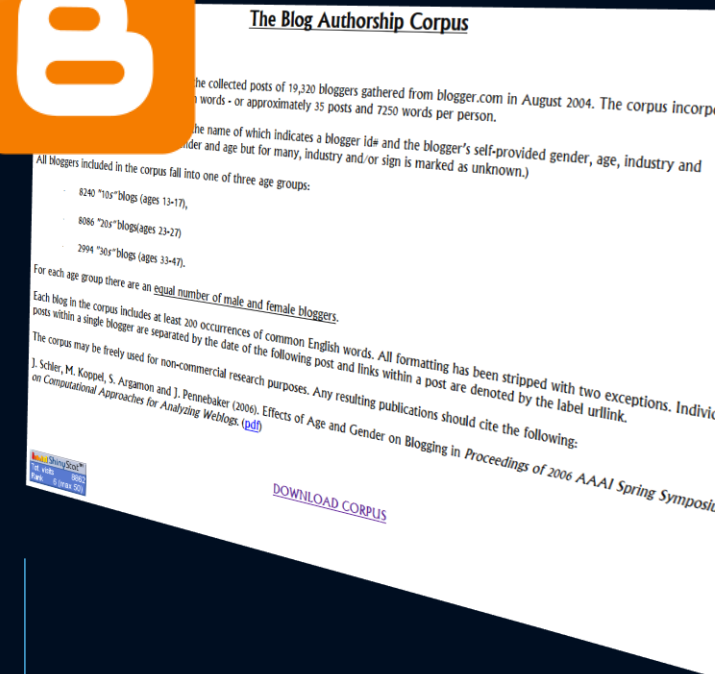
Methods of authorship attribution

- **Similarity-based paradigm**
 - more appropriate for many candidate authors
 - measure distance between anonymous and known text
 - Attribution to most similar one



Corpus and data usage

- 10,000 blogs (blogger.com, August 2004)
 - balanced for gender
 - mainly written in English
- Usage:
 - 2,000 words of known text of each blog
 - Snippet of 500 words
 - Task: **By which of the candidate authors (if any) was the snippet written?**



A recollection of the corpus can be found on:
<http://u.cs.biu.ac.il/~koppel/ BlogCorpus.htm>

Approach

Experiment


Implementation

Results

Preparing the data:

Space-free character 4-grams

- String of characters...
 - of length 4, includes no spaces
 - of ≤ 4 characters, surrounded by spaces
- 250,000 unique, overlapping SFC₄G's
- measurable in any language, no background knowledge needed



Hello world!

Approach

Experiment

Implementation

Results

Preparing the data:

Space-free character 4-grams

- String of characters...
 - of length 4, includes no spaces
 - of ≤ 4 characters, surrounded by spaces

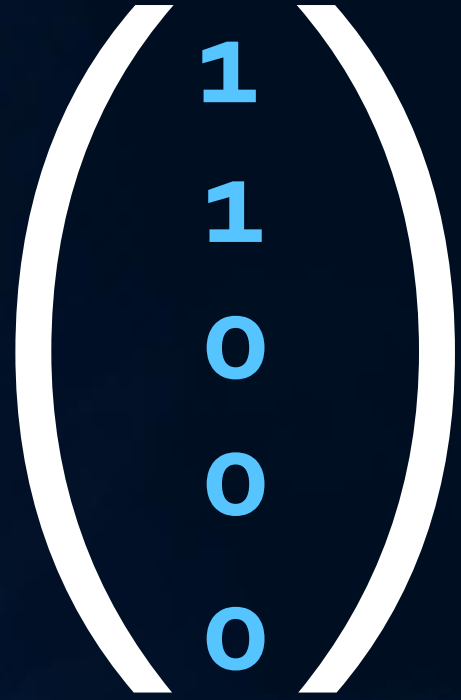
Hell **worl**
ello **orld**
 rld!



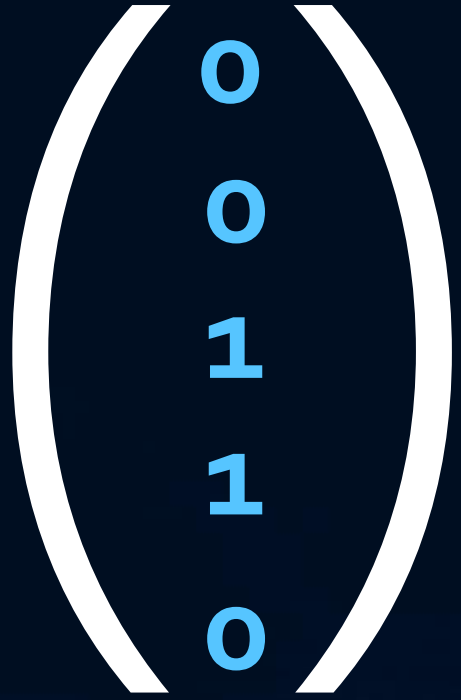
Preparing the data:

Generating feature sets

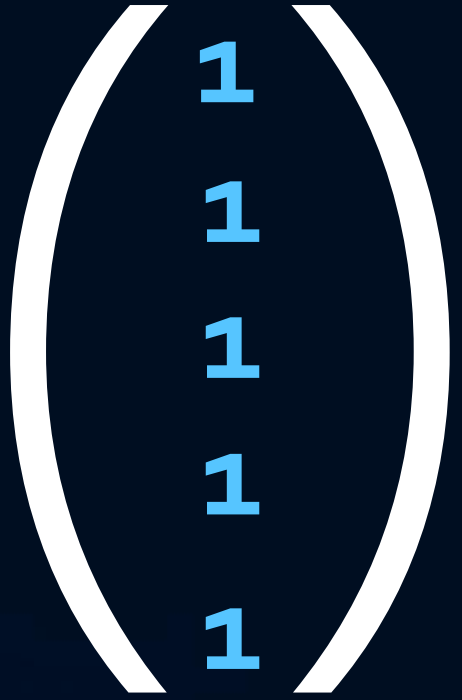
Hell
 ello
 worl
 orld
 rld!



"Hello"



"world"



"Hello world!"

Approach

Experiment

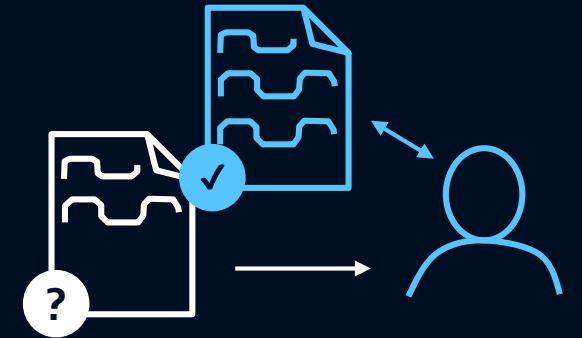
Implementation

Results

The main algorithm

Assumptions

- Known text of snippet's author
=
text most similar to the snippet
- Also if feature set is varied



The main algorithm

Pseudocode

```
repeat ( $k_1$  times) {  
    randomly choose fraction  $k_2$  of feature set  
    find top mach using cosine similarity  
}  
for each (candidate author  $A$ ) {  
    score( $A$ ) = proportion of times  $A$  is top match  
}  
Output:  
if (  $\max \text{score} (A) > \sigma^*$  ) {  $\arg \max_A \text{score} (A)$  }  
else { Don't know! }
```

Given:

- Snippet of length L_1
- Known-texts of length L_2 for each of C candidates

The main algorithm

Pseudocode

```
repeat ( $k_1$  times) {  
    randomly choose fraction  $k_2$  of feature set  
    find top mach using cosine similarity  
}  
for each (candidate author  $A$ ) {  
    score( $A$ ) = proportion of times  $A$  is top match  
}  
Output:  
if (  $\max \text{score} (A) > \sigma^*$  ) {  $\arg \max_A \text{score} (A)$  }  
else { Don't know! }
```

Parameters:

k_1 : number of different feature sets used

k_2 : fraction of possible features of the set

A : a candidate author

σ^* : threshold to be reached by a candidate autor

Approach

Experiment

Implementation

Results

Steps of reproduction / implementation

- pre-process corpus (✓) original corpus not available!
- read in data from files ✓
- extract plain text (✓) some improvements pending...
- generate space-free character 4-grams ✓
- generate feature sets ! current algorithm too slow!
- run main algorithm ToDo

Approach

Experiment

Implementation

Results

1) Pre-process corpus (✓)

Original corpus	Corpus from Prof. Koppel's website
collected in 2004	recollected later
10,000 blogs	19,320 blogs
all blogs mainly written in English	also contains blogs in Chinese, etc.

- Eliminated blogs with not enough text, ignored non-English characters
- included all space-free 4-grams as well as all *words of length < 4*
- only used k most frequent features in corpus

2) Read data from files ✓

Java code

```
public ArrayList <String> generate4gramLists () {
    File [] allNames = getFileNames();
    TreeSet <String> fourGrams = new TreeSet <> ();
    // LOOP 1: Repeat for each file...
    for (int i = 0; i < allNames.length; i++) {
        try {
            // Read from blog file and generate file with the same name to save 4grams
            br = new BufferedReader(...);
            pw = new PrintWriter (...);
            String line; // Saves a single line read in by the buffered reader
            boolean isPost = false;
            // LOOP 2: Read blog file line by line until end of document is reached...
            while ((line = br.readLine()) != null) {
                ...
                ...
            }
        }
    }
}
```


3) Extract plain text (✓)

Java code

```
if (line.startsWith("<post>")) {
    isPost = true;
    line = ""; // Reset so <post> is not added
}
if (line.endsWith("</post>")) {
    isPost = false;
}
// Remove non-content from line if it is between
if (isPost) {
    ...
    // Generate Array that contains all words (
    String[] words = line.split(" ");
    // LOOP 3: Repeat for each word
    int l = words.length;
    for (int j = 0; j < l; j++) {
        ...
        ...
    }
}
```

Example: XML file from blog corpus

```
<date>28, June, 2004</date>
<post>
    Finally! Some colour to my posts.. <a href="#">urlLink</a>
</post>
<date>27, June, 2004</date>
<post>
    I'm turning nocturnal from all the late nights w
    feeling it sting. Horrible thing is school reope
    if I can wake up on time or stay awake throughou
</post>
```

4) Generate 4-grams ✓

Java code

```
for (int j = 0; j < l; j++) {
    if (words[j].length() <= 4) {
        // Add 4-gram to tree and write into 4-gram file of this blog
        if (!words[j].equals("")) {
            fourGrams.add(words[j]);
            pw.print(words[j]+" ");
            pw.println(""); // Go to next line in 4-gram file of this blog
        }
    }
    else {
        // For example: "Hello" has length of 5
        // So loop is repeated 2 times (at the second time k=1 <= 1 = 5-4)
        // At the first time the substring is "Hell" (char 0-4)
        // and at the second time it is "ello" (char 1-5)
        for (int k = 0; k <= words[j].length() - 4; k++) {
            fourGrams.add(words[j].substring(k, k+4));
            pw.print(words[j].substring(k, k+4)+" ");
        }
    }
}
```

Approach

Experiment

Implementation

Results

4) Generate 4-grams ✓

File that contains all space-free character 4-grams

```
ada_  
adaa  
adab  
adac  
adad  
adae  
adaf  
adag  
adah  
adai  
adaj  
adak  
adal  
adam  
adan  
adao
```

4-gram file of one single blog

```
Slas lash ashd shdo hdot  
rais aise ises  
lots  
of  
inte nter tere eres rest esti stin ting  
thou houg ough ught ghts  
abou bout  
bann anne nner  
ads  
.  
The  
idea  
is  
to
```

5) Generate feature sets !

Pseudo code

```
load all space-free character 4-grams from file into list L1
load 4-grams of last 500 words from random blog (snippet) to list L2
generate feature set for snippet {
  declare array A1 of the length of L1 and fill with 0's
  for each 4-gram in L1 {
    for each 4-gram in L2 {
      if (4-grams from arrays of both loops match) {
        increase value of A1 at the position of iterator of outer loop by 1
      }
    }
  }
}
save A1 to file
repeat steps feature set generation similarly for each of the 10,000 known texts
```

Approach

Experiment

Implementation

Results

5) Generate feature sets !

Pseudo code

```
load all space-free character 4-grams from file into list L1
load 4-grams of last 500 words from random blog (snippet) into list L2
generate feature set for snippet {
  declare array A1 of the length of L1 and fill with 0's
  for each 4-gram in L1 {
    for each 4-gram in L2 {
      if (4-grams from arrays of both loops match) {
        increase value of A1 at the position of iterator of outer loop by 1
      }
    }
  }
}
save A1 to file
repeat steps feature set generation similarly for each of the 10,000 known texts
```

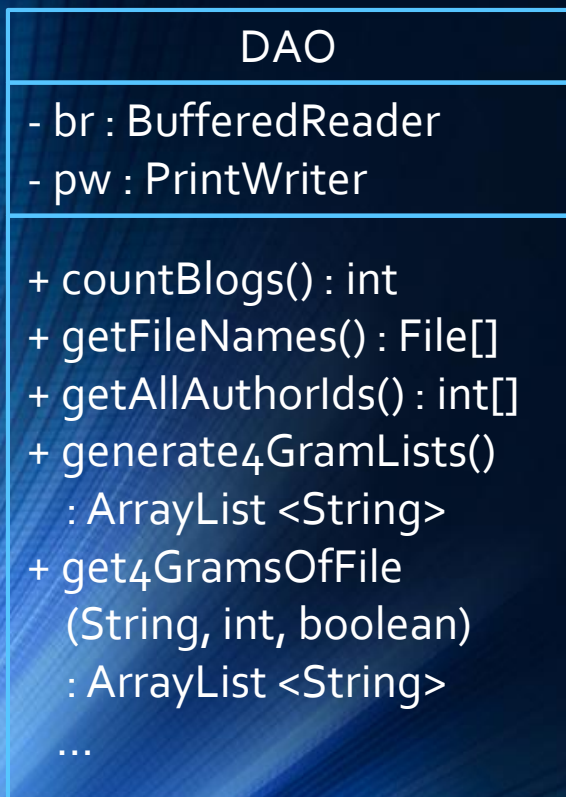
10,000 (number of blogs)
* 250,000 (all possible 4-grams)
* 100,000 (4-grams per blog)
= 250,000,000,000,000



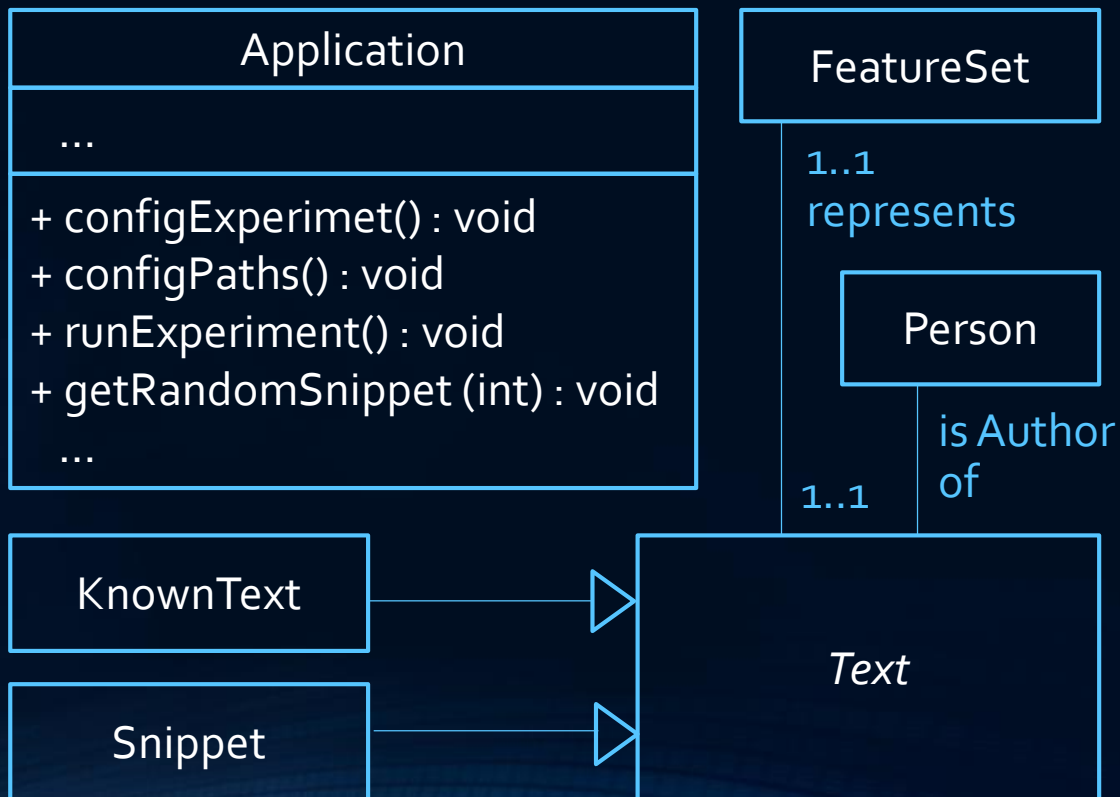
Software architecture



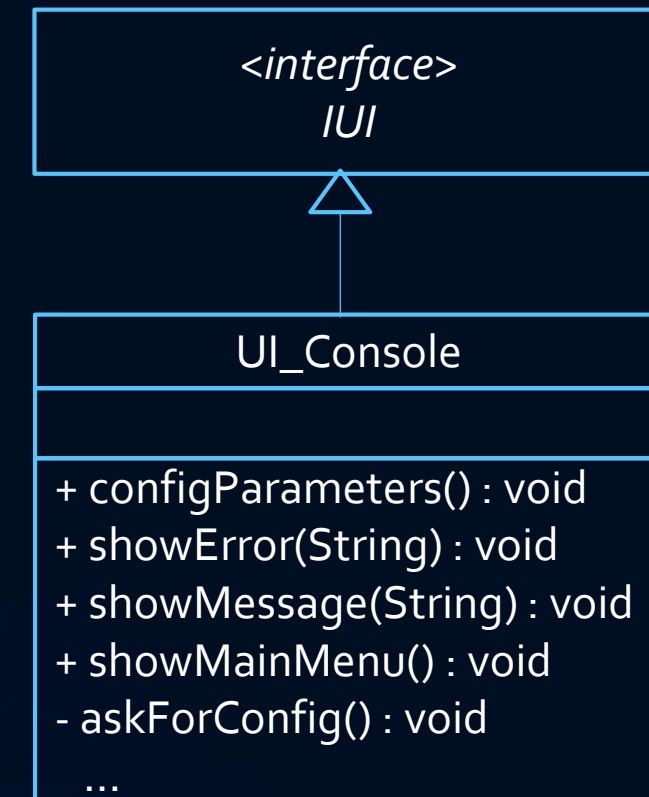
Data Access



Programm logic



User interface



Approach

Experiment

Implementation

Results

Parameters

k_1 : number of different feature sets used

k_2 : fraction of possible features per iteration

C : number of candidate authors

σ^* : threshold to be reached by a candidate autor

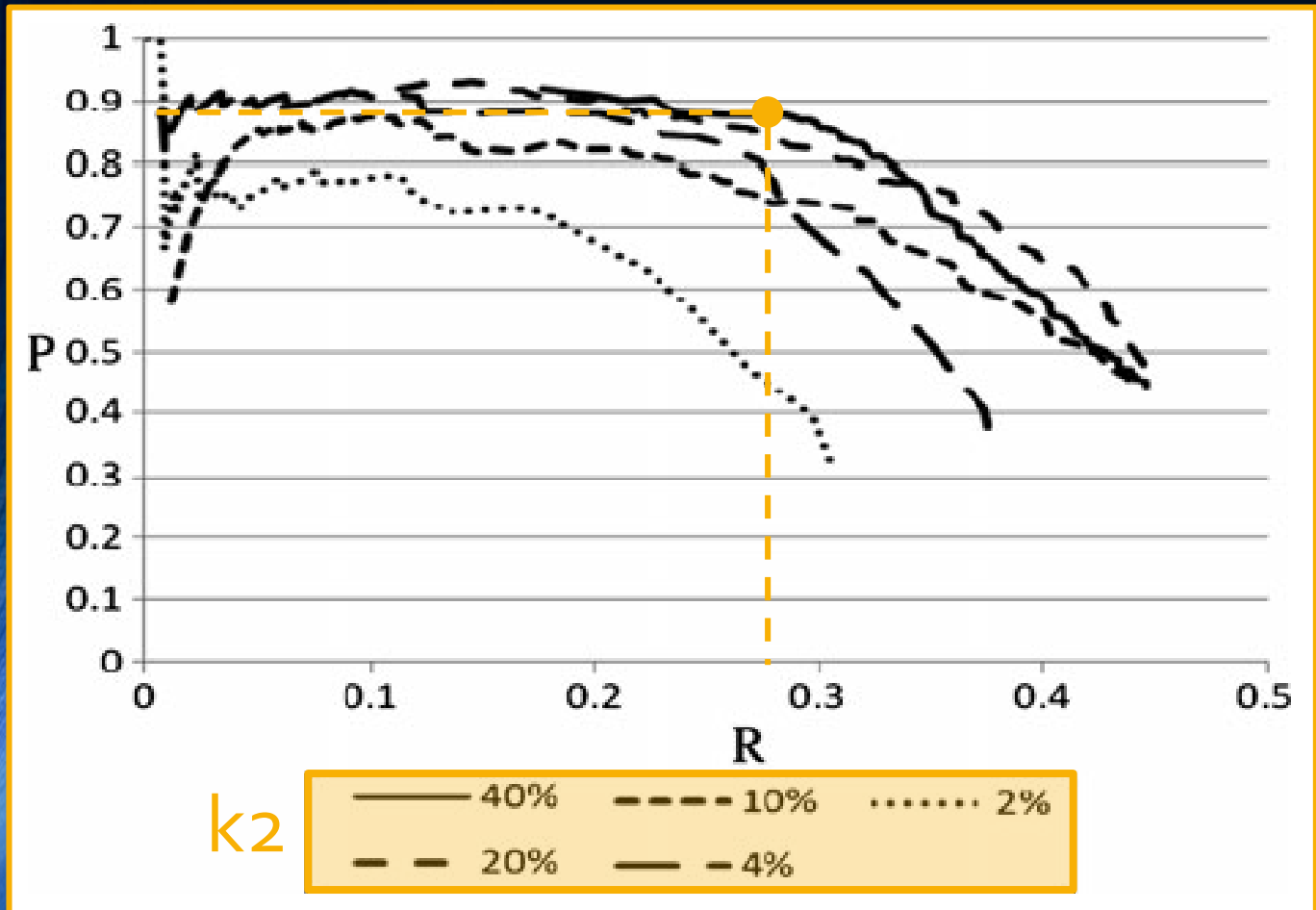
Approach

Experiment

Implementation

Results

k_2 : fraction of features per iteration

 $k_1 = 100$

(Iterations)

 $\sigma^* = 90\%$

(Threshold)

 $C = 10,000$

(Candidate authors)

 $L_1 = 500$

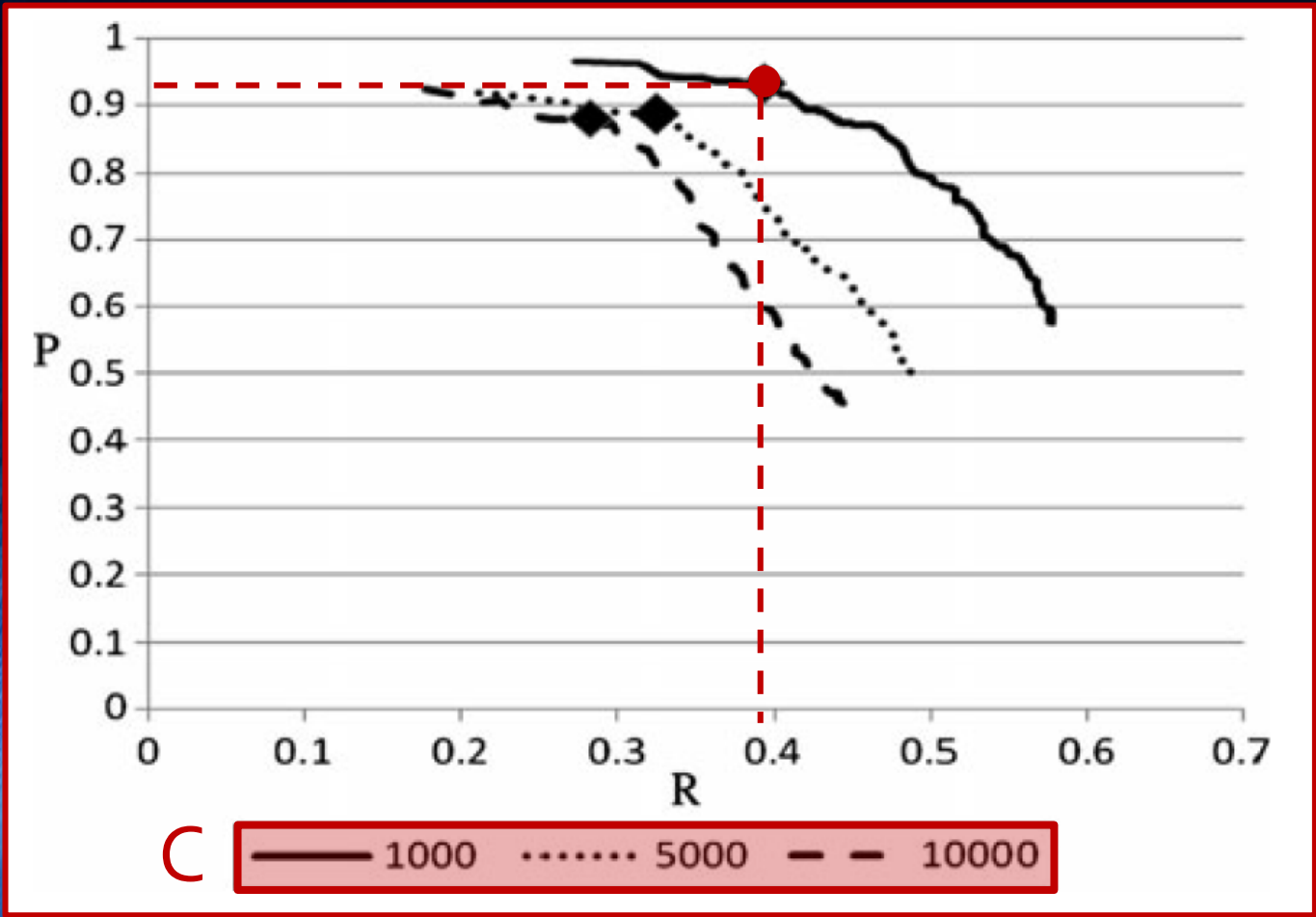
(Snippet length)

 $L_2 = 2,000$

(Known-text length)

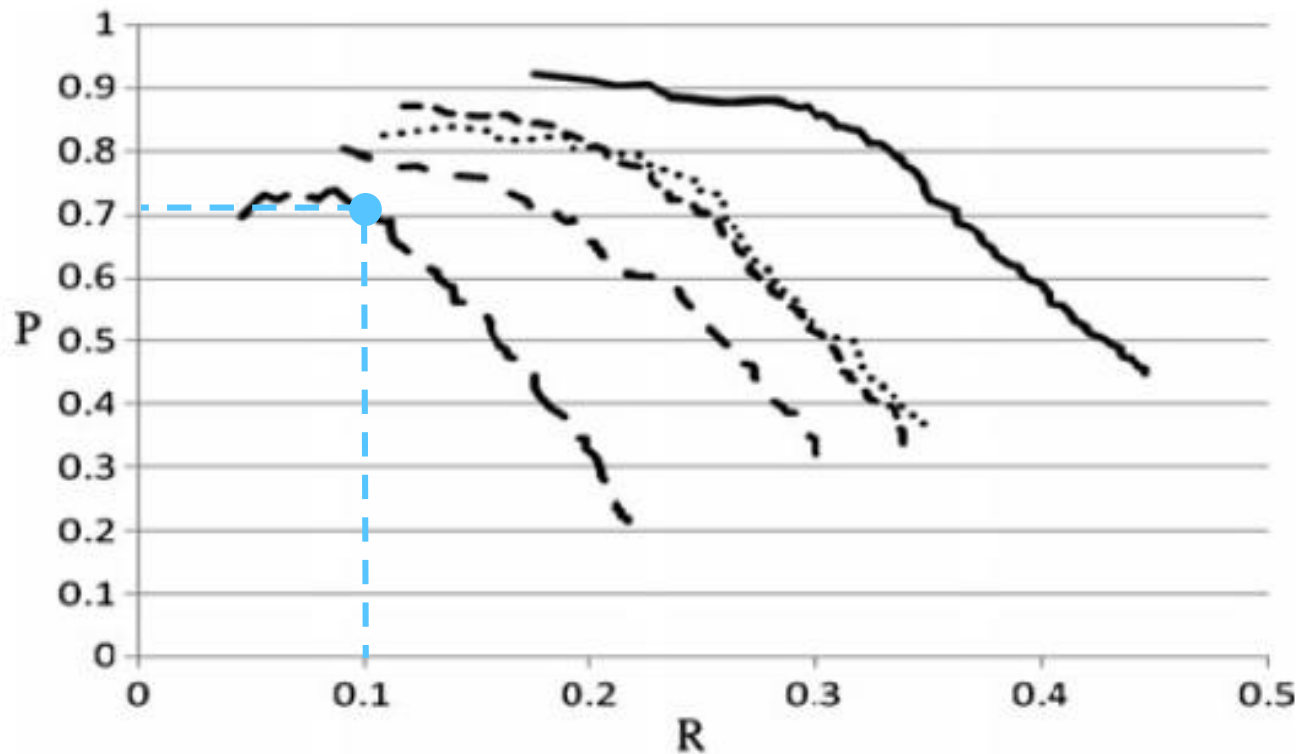


C: number of candidate authors

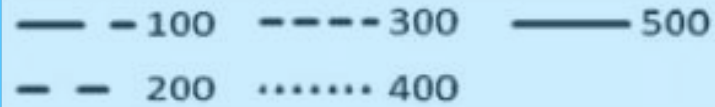


- $k_1 = 100$ (Iterations)
- $k_2 = 40\%$ (Features / iteration)
- $\sigma^* = 90\%$ (Threshold)
- $L_1 = 500$ (Snippet length)
- $L_2 = 2,000$ (Known-text length)

L1: Snippet length



L1



$k_1 = 100$

(Iterations)

$k_2 = 40\%$

(Features / iteration)

$\sigma^* = 90\%$

(Threshold)

$C = 10,000$

(Candidate authors)

$L_2 = 2,000$

(Known-text length)

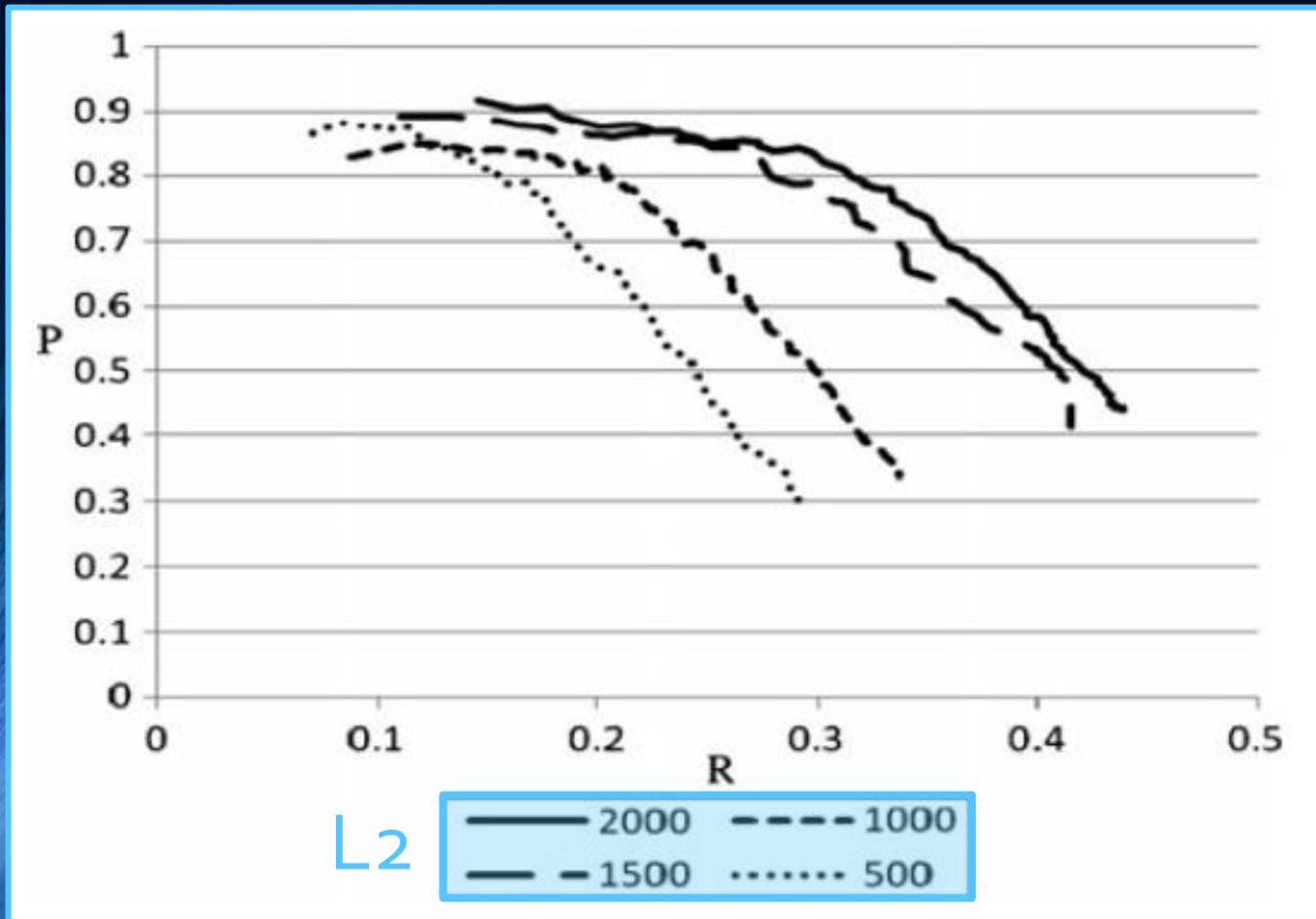
Approach

Experiment

Implementation

Results

L2: Known-text length

 $k_1 = 100$

(Iterations)

 $k_2 = 40\%$

(Features / iteration)

 $\sigma^* = 90\%$

(Threshold)

 $C = 10,000$

(Candidate authors)

 $L_1 = 500$

(Snippet length)

Approach

Experiment

Implementation

Results

Abstract: Extension of the algorithm

- Useful for forensic applications:
Estimated probability that assigned author is actual one
 - Precision and coverage can be predicted using regression
 - Probability p :
 - Probability that author is in candidate set
 - Value [0 - 1.0] provided by user
 - E : Probability that some author is assigned when the actual one is not in the candidate set
 - $\frac{p * H * P}{p * H + (1 - p) * E}$ = Probability assigned author is the actual one

Approach

Experiment

Implementation

Results

Conclusions (Paper)

- Simple similarity-based method can solve even difficult problems
 - many candidate authors
 - limited length of known-text
 - actual author might not be in candidate set
- Passable results even for snippets of only 100 words
- Method not useful for small open candidate sets and limited anonymous text

Approach

Experiment

Implementation

Results

Conclusions (Reproducibility)

- ✓ Main algorithm well described
- ✓ Corpus the experiment was performed with available on the author's website...
- ! ...but a recollection and not the original one – not similar to the original one in some points
- ! No description how the vectors are generated from the 4-grams or how they are handled effectively within the program
- ! Plugin Apache Lucene used for indexing not mentioned in paper