Avoiding cheating with automated plagiarism detection

Rune Borge Kalleberg
Developer @ Lærelyst AS

Thesis Overview

“Towards Detecting Textual Plagiarism Using Machine Learning Methods”

Compare suspicious and source documents

Calculate similarity using some metric(s)

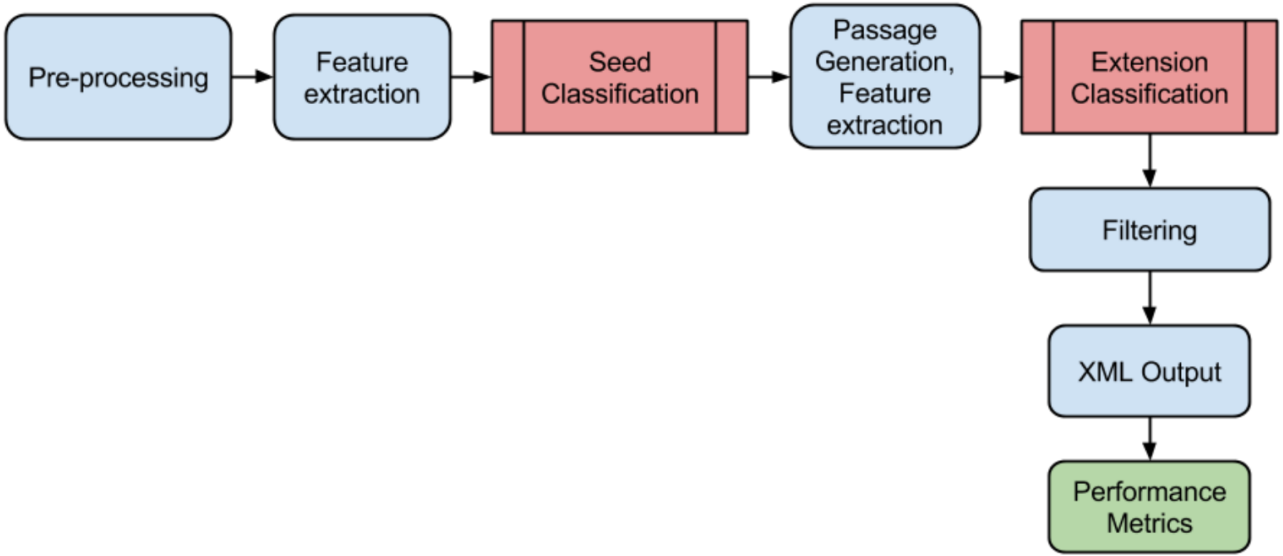
Used machine learning to detect and classify plagiarism

Used existing metrics and some novel methods

Decision Trees for classification

Compared against State of the Art(PAN @ CLEF) using common data set.

Methodology Overview



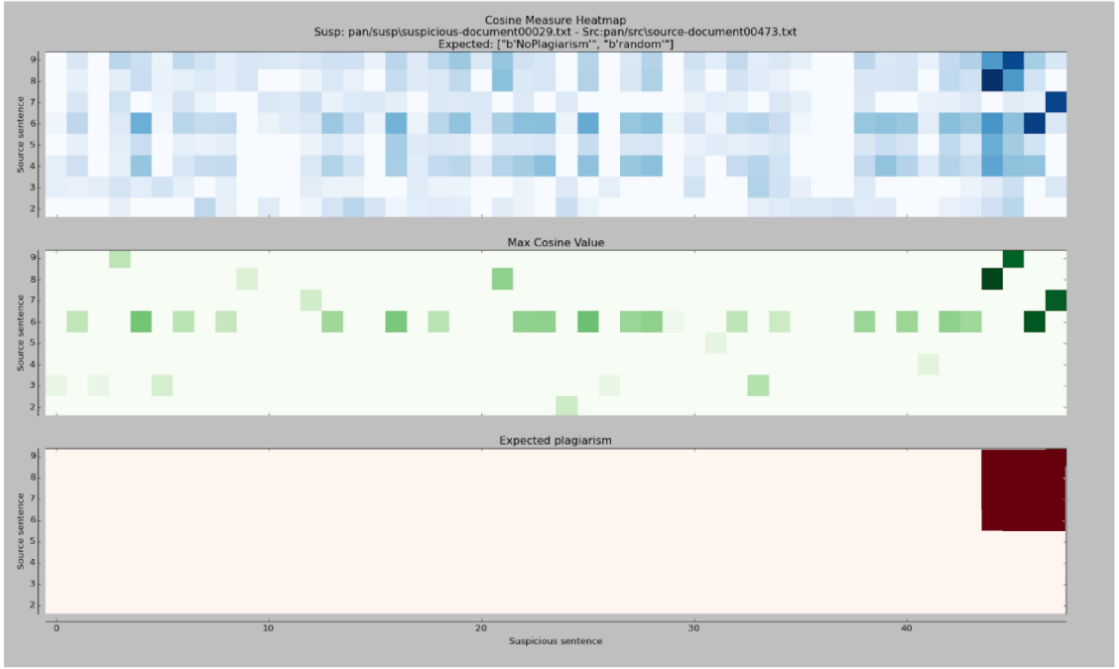
Tokenization models

N-Grams

- Words
- Characters
- Sentences
- Stop words
- Skip-n-grams

Bag of Words

- Word frequencies in sentences



Pre-processing and Seeding

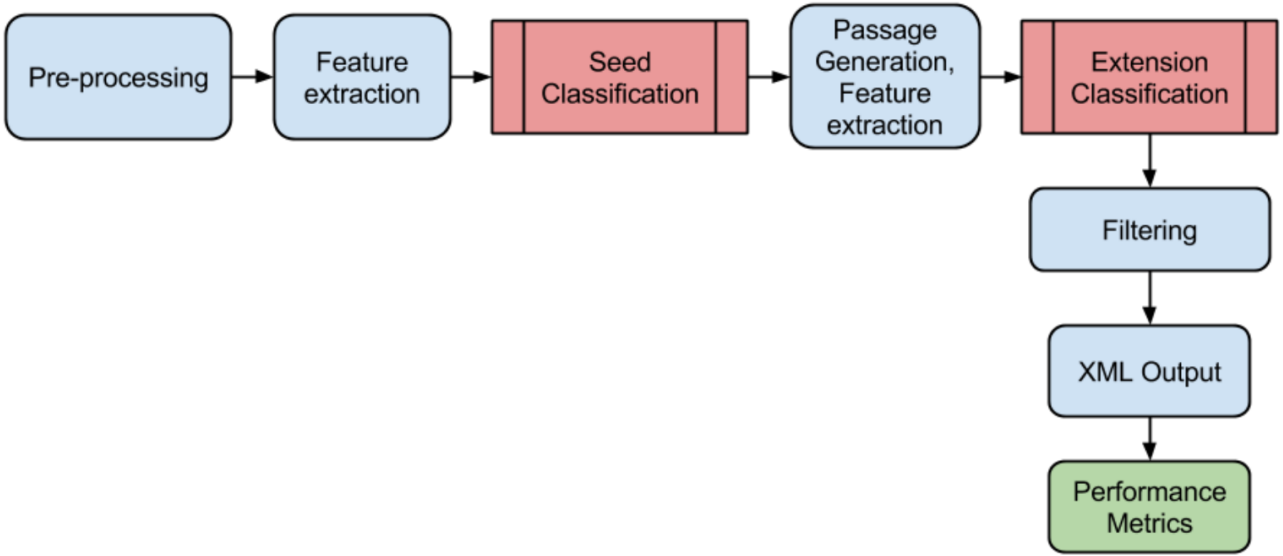
Stop word removal

Stemming

Extract features from BoW

Feature	Value	Description
Cosine	Float	BoW similarity measurement. Frequency dependent
Dice Coefficient	Float	BoW similarity measurement. Frequency invariant
IsMax(Cos/Dice)	Boolean	Returns TRUE if it is source fragment with the highest Cos/Dice value for that suspicous fragment
MaxDiff(Cos/Dice)	Float	Difference between the current source fragment and the source fragment with the highest maximum similarity value of the same type for that suspicious fragment.
MeanDiff(Cos/Dice)	Float	Difference between the current source fragment and the document mean maximum values of the same type for that suspicious fragment.
MaxNeighbour(Cos/Dice)	Float	The highest value of the given similarity type for immediate suspicious fragment neighbours.
VerticalMaxDist(Cos/Dice)	Integer	Distance in fragments between the current source fragment and the fragment with the highest similarity value of the given type.
SrcSuspLenRatio	Float	The length ratio between the source and suspicious passages.

Methodology Overview



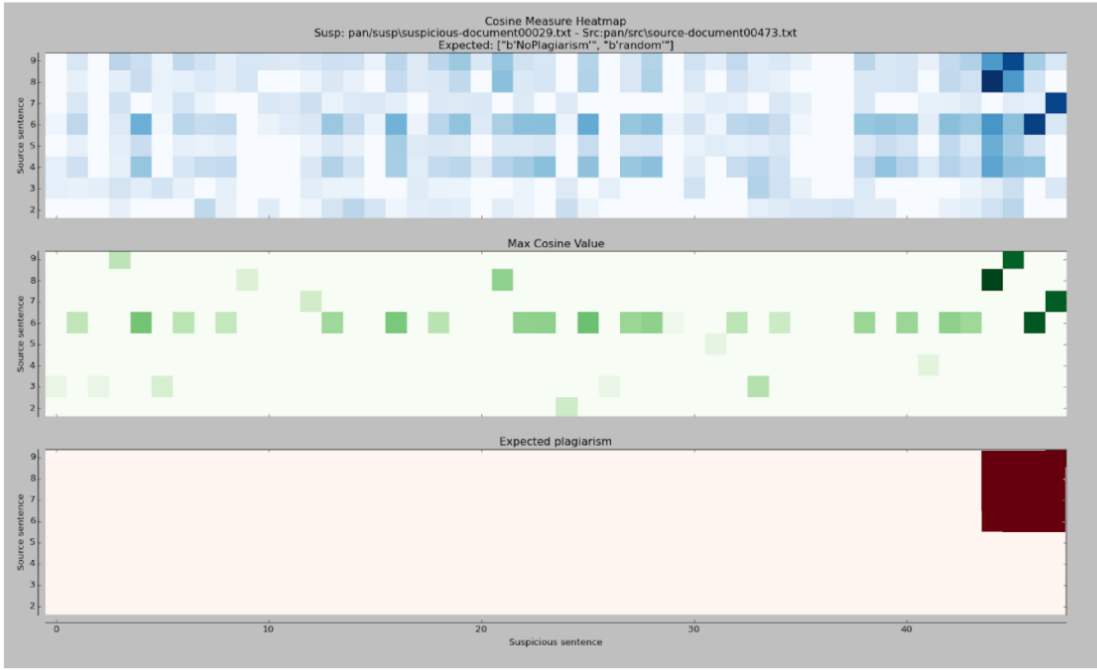
Passage Generation

Group horizontally adjacent seeds

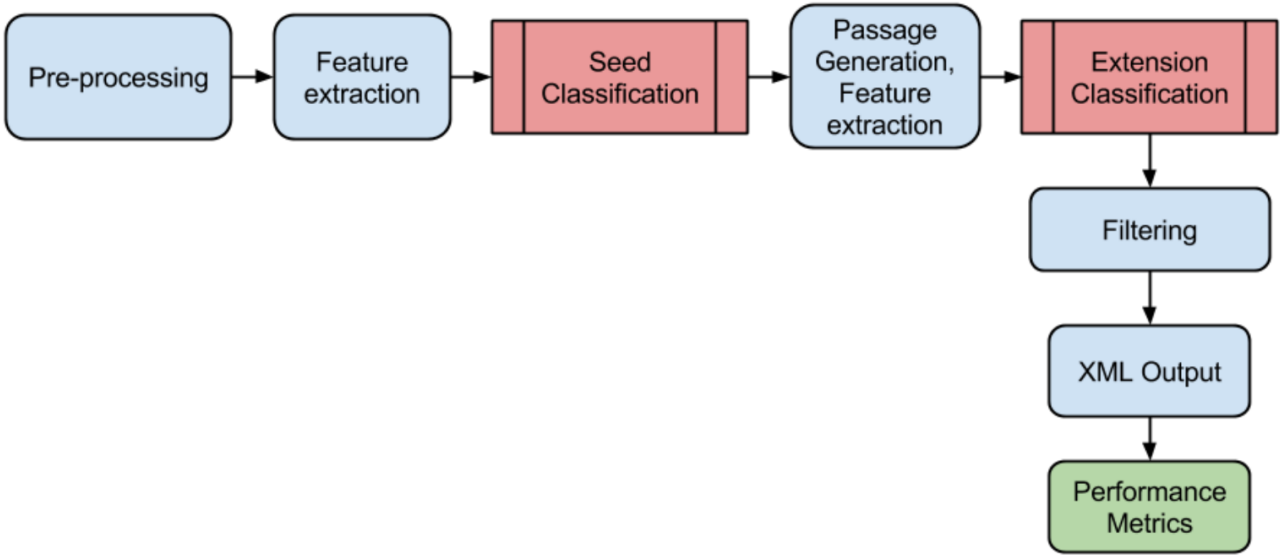
Exhaustive pairing of groups as passages

Extract features from passages

Use these features in classification



Methodology Overview



How can this be used in source code?

Code is a textual language as any other

Different N-gram models

Bag of Symbols?

Possible to analyze code symbolically/structurally?

Most programming is highly sequential.

“Sentences” can’t be moved and still give the text the same meaning!

Example!

Task: Find the sum of all the multiples of 3 or 5 below 1000.

```
mySum = sum([i for i in range(1000) if (i % 3 == 0 or i % 5 == 0)])
```

Processed:

```
s = f ( s for s in f ( 1000 ) if ( s % 3 == 0 or s % 5 == 0 ) ] )
```

```
sum = 0
for n in range(1000):
        if(n % 3 == 0):
         sum++
        else if(0 == n % 5 ):
         sum++
```

Processed:

```
s = 0 for s in f ( 1000 ) : if ( s % 3 == 0): s ++ else if ( 0 == s % 5 ) : s ++
```

Common consecutive 3+-grams(varying sizes for simplicity)

“for s in f (1000)”,

“if (s % 3 == 0”,

”s % 5”

s = f (s for s in f (1000) if (s % 3 == 0 or s % 5 == 0)])

s = 0 for s in f (1000) : if (s % 3 == 0): s ++ else if (0 == s % 5) : s ++

Useful sources for study

M. Potthast, et. al, “Overview of the 6th International Competition on Plagiarism Detection.” 2014. [Online]. Available: http://www.uni-weimar.de/medien/webis/publications/papers/stein 2014k.pdf

Towards Detecting Textual Plagiarism Using Machine Learning Methods - Rune Borge Kalleberg